

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL  
ESCOLA DE ENGENHARIA  
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

VICENTE COSTAMILAN DA CUNHA

**Métodos de Segmentação Automática de Sinais  
de Eletromiografia de Superfície para  
Classificação de Movimentos Utilizando Redes  
Neurais Artificiais**

Porto Alegre

2015

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL  
ESCOLA DE ENGENHARIA  
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

VICENTE COSTAMILAN DA CUNHA

**Métodos de Segmentação Automática de Sinais de  
Eletromiografia de Superfície para Classificação de  
Movimentos Utilizando Redes Neurais Artificiais**

Projeto de Diplomação apresentado ao Departamento de Engenharia Elétrica da Escola de Engenharia da Universidade Federal do Rio Grande do Sul, como requisito parcial para Graduação em Engenharia Elétrica

Orientador: Prof. Dr. Eng. Alexandre Balbinot

Porto Alegre

2015

## CIP - Catalogação na Publicação

Cunha, Vicente Costamilan da  
Métodos de Segmentação Automática de Sinais de  
Eletromiografia de Superfície para Classificação de  
Movimentos Utilizando Redes Neurais Artificiais /  
Vicente Costamilan da Cunha. -- 2015.  
77 f.

Orientador: Alexandre Balbinot.

Trabalho de conclusão de curso (Graduação) --  
Universidade Federal do Rio Grande do Sul, Escola de  
Engenharia, Curso de Engenharia Elétrica, Porto  
Alegre, BR-RS, 2015.

1. Eletromiografia de Superfície. 2. Segmentação de  
Sinais. 3. Redes Neurais Artificiais. I. Balbinot,  
Alexandre, orient. II. Título.

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL  
ESCOLA DE ENGENHARIA  
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

VICENTE COSTAMILAN DA CUNHA

**Métodos de Segmentação Automática de Sinais de  
Eletromiografia de Superfície para Classificação de  
Movimentos Utilizando Redes Neurais Artificiais**

Projeto de Diplomação apresentado ao Departamento de Engenharia Elétrica da Escola de Engenharia da Universidade Federal do Rio Grande do Sul, como requisito parcial para Graduação em Engenharia Elétrica

Trabalho aprovado. Porto Alegre, 03 de dezembro de 2015:

---

**Prof. Dr. Eng. Alexandre Balbinot**  
Orientador

---

**Prof. Dr. Eng. Altamiro Susin**  
Convidado

---

**M.<sup>a</sup> Eng. Gabriela Favieiro**  
Convidada

---

**Eng. Vinicius Cene**  
Suplente

*A Gilberto, mi padre, torre de razón y de firme fe;  
e a todos aqueles que tomarem interesse neste estudo.*

# Agradecimentos

Aos demais colaboradores e pesquisadores do laboratório de Instrumentação Eletro-Eletrônica, em especial Vinicius Cene e Fernanda Trevisol, que desenvolveram a coleta da base de dados utilizada e prestaram auxílio de forma geral.

*Take nothing on its looks;  
take everything on evidence.  
There's no better rule.*

Charles Dickens, Great Expectations

# Resumo

A segmentação de sinais de eletromiografia (EMG) é parte essencial da etapa de pré-processamento em aplicações de reconhecimento de movimentos e controle de próteses. Métodos de segmentação automática possibilitam a individualização de trechos de interesse do sinal correspondentes a esforços musculares e o descarte de trechos de sinal com baixa atividade muscular, por exemplo. Neste estudo, quatro métodos para segmentação automática de sinais de EMG de superfície, adaptados de outros trabalhos na área, foram propostos e implementados em MATLAB R2015a. Os métodos foram aplicados nos sinais da base de dados número 2 do projeto NinaPro (*Non-Invasive Adaptive Prosthetics project*) e nos sinais da base de dados adquiridos pelo Laboratório de Instrumentação Eletro-Eletrônica da UFRGS. Redes neurais artificiais (RNA) foram utilizadas para classificar os sinais segmentados com os quatro métodos de segmentação entre 17 diferentes movimentos de mão e punho. Resultados mostram ser possível utilizar segmentação automática baseada na detecção de picos com *threshold* para resultados de valor *F* médio de classificação acima de 91% entre diferentes classes de movimentos.

**Palavras-chave:** Eletromiografia de Superfície. Segmentação de Sinais. Redes Neurais Artificiais.

# Abstract

EMG signal segmentation is a key part of preprocessing in movement identification applications and prostheses control. Automatic segmentation methods allow for the separation of signal segments of interest due to muscular effort while discarding signal related to low muscular activity, for example. In this study, four different automatic segmentation methods for surface EMG signal are proposed, adapted from other works in the area, and written for MATLAB R2015a. The methods were applied to signals in the second NinaPro (Non-Invasive Adaptive Prosthetics project) database and signals that were acquired at the Electronic Instrumentation Laboratory (IEE) at UFRGS. Artificial neural networks (ANN) were designed for the classification of segmented signals with the four segmentation methods between 17 different hand and wrist movements. Results show it is possible to achieve mean classification  $F$  values above 91% throughout different single movement classes using automatic segmentation method based in constant threshold peak detection.

**Keywords:** Surface Eletromiography. Signal Segmentation. Artificial Neural Networks.

# Listas de Figuras

Figura 1 – Soma de potenciais de ação das $n$ fibras de uma unidade motora, formando uma MUAP $h(t)$ .	16
Figura 2 – MUAPTs de diferentes MUs somam-se para compor o sinal adquirido por um canal de EMG.	17
Figura 3 – Fluxograma representativo do MTD1.	19
Figura 4 – Fluxograma representativo do MTD2.	21
Figura 5 – Fluxograma representativo do MTD3.	26
Figura 6 – Fluxograma representativo do MTD4.	27
Figura 7 – Diagrama representativo do modelo de um neurônio artificial genérico.	28
Figura 8 – Exemplo de uma RNA em arquitetura <i>feedforward</i> .	28
Figura 9 – Posicionamento de eletrodos de superfície no braço de um voluntário.	30
Figura 10 – Diagrama de blocos para o sistema de aquisição da base de dados do IEE.	31
Figura 11 – Cenas do vídeo apresentado aos voluntários na aquisição do IEE.	32
Figura 12 – Diagrama de blocos geral para os métodos de segmentação.	32
Figura 13 – Exemplo para retificação e normalização de trecho de sinal de EMG.	34
Figura 14 – Diagrama de blocos para processo de classificação de movimentos.	36
Figura 15 – Estrutura de RNAs utilizadas.	38
Figura 16 – Média e moda do número de segmentos obtidos por classe de movimento e valor $F$ médio de RNAs com diferentes combinações de parâmetros. Método de segmentação: MTD1.	42
Figura 17 – Média e moda do número de segmentos obtidos por classe de movimento e valor $F$ médio de RNAs com diferentes combinações de parâmetros. Método de segmentação: MTD2.	43
Figura 18 – Média e moda do número de segmentos obtidos por classe de movimento e valor $F$ médio de RNAs com diferentes combinações de parâmetros. Método de segmentação: MTD3.	44
Figura 19 – Média e moda do número de segmentos obtidos por classe de movimento e valor $F$ médio de RNAs com diferentes combinações de parâmetros. Método de segmentação: MTD4.	45
Figura 20 – Valor $F$ médio por classe de movimento para cada base de dados, utilizando parâmetros selecionados em cada método de segmentação.	46

# **Lista de Tabelas**

Tabela 1 – Parâmetros utilizados para definir o MTD1.	18
Tabela 2 – Parâmetros utilizados para definir o MTD2.	20
Tabela 3 – Parâmetros utilizados para definir o MTD3.	22
Tabela 4 – Parâmetros utilizados para definir o MTD4.	22
Tabela 5 – Exemplo de matriz de confusão.	25
Tabela 6 – Lista de movimentos de interesse apresentados aos voluntários.	33
Tabela 7 – Parâmetros ajustáveis para os métodos de segmentação.	35
Tabela 8 – Combinações de parâmetros utilizados nos métodos de segmentação.	39
Tabela 9 – Coeficientes de correlação entre parâmetros utilizados e média de número de segmentos obtido por classe de movimento.	40
Tabela 10 – Combinações de parâmetros selecionados para cada base de dados.	41
Tabela 11 – Valor $F$ médio por classe de movimento utilizando parâmetros selecio- nados em cada método	47

# **Lista de Abreviaturas e Siglas**

BEP	<i>Beginning Extraction Point</i>
DAQ	<i>Data Acquisition</i>
DBSCAN	<i>Density-Based Spatial Clustering of Applications with Noise</i>
EEP	<i>Ending Extraction Point</i>
EMG	Eletromiografia
EMGs	Eletromiografia de Superfície
IEE	Laboratório de Instrumentação Eletro-Eletrônica
MU	<i>Motor Unit</i>
MUAP	<i>Motor Unit Action Potencial</i>
MUAPT	<i>Motor Unit Action Potencial Trains</i>
MTD#	<i>Método Número #</i>
NI	<i>National Instruments</i>
NinaPro	<i>Non-Invasive Adaptive Prosthetics project</i>
RNA	Rede Neural Artificial

# Sumário

<b>1</b>	<b>INTRODUÇÃO</b>	<b>14</b>
<b>2</b>	<b>REVISÃO BIBLIOGRÁFICA</b>	<b>16</b>
<b>2.1</b>	<b>Sinais de Eletromiografia</b>	<b>16</b>
<b>2.2</b>	<b>Métodos de Segmentação</b>	<b>17</b>
2.2.1	Método 1 - método iterativo utilizando <i>thresholding</i> para detecção de centros de segmentos de comprimento constante (MTD1)	18
2.2.2	Método 2 - método não iterativo utilizando <i>thresholding</i> para detecção de centros de segmentos de comprimento constante (MTD2)	20
2.2.3	Método 3 - método com janela deslizante para detecção de BEP e EEP de segmentos utilizando variação total (MTD3)	20
2.2.4	Método 4 - método com janela deslizante para detecção de BEP e EEP de segmentos utilizando <i>thresholding</i> (MTD4)	22
<b>2.3</b>	<b>Princípios Básicos sobre Redes Neurais Artificiais</b>	<b>23</b>
2.3.1	Modelo Matemático de RNAs	23
2.3.2	Treinamento de RNAs	24
2.3.3	Valor <i>F</i>	24
<b>3</b>	<b>METODOLOGIA EXPERIMENTAL</b>	<b>29</b>
<b>3.1</b>	<b>Bases de Dados Utilizadas</b>	<b>29</b>
3.1.1	Posicionamento de Eletrodos	29
3.1.2	Sistema para Aquisição de EMGs no IEE	29
3.1.3	Movimentos de Interesse	30
<b>3.2</b>	<b>Métodos de Segmentação</b>	<b>31</b>
3.2.1	Preprocessamento	31
3.2.2	Parâmetros Ajustáveis	32
3.2.3	Agrupamento das Posições de Segmentos de Diferentes Canais com DBSCAN	34
<b>3.3</b>	<b>Classificação de Segmentos com Redes Neurais Artificiais</b>	<b>36</b>
3.3.1	Características Utilizadas como Preditores	36
3.3.2	Classes de Movimentos Utilizadas como Resposta	37
3.3.3	Separação de Grupos de Treino, Validação e Teste para Treinamento	37
3.3.4	Estrutura de RNAs Treinadas	38
<b>4</b>	<b>RESULTADOS E DISCUSSÕES</b>	<b>39</b>
<b>4.1</b>	<b>Segmentação e Classificação por Combinacões de Parâmetros</b>	<b>39</b>
<b>4.2</b>	<b>Segmentação e Classificação por Classes de Movimento</b>	<b>41</b>

5	CONCLUSÕES . . . . .	48
6	PROPOSTAS DE TRABALHOS FUTUROS . . . . .	49
	REFERÊNCIAS BIBLIOGRÁFICAS . . . . .	50
	<b>APÊNDICES</b>	<b>53</b>
	APÊNDICE A – FUNÇÃO EM MATLAB PARA MTD1 . . . . .	54
	APÊNDICE B – FUNÇÃO EM MATLAB PARA MTD2 . . . . .	57
	APÊNDICE C – FUNÇÃO EM MATLAB PARA MTD3 . . . . .	60
	APÊNDICE D – FUNÇÃO EM MATLAB PARA MTD4 . . . . .	63
	APÊNDICE E – CÓDIGO EM MATLAB PARA SEGMENTAÇÃO DE SINAIS, TREINO DE RNA E RESULTADOS DE CLASSIFICAÇÃO . . . . .	66
	APÊNDICE F – CÓDIGO UTILIZADO PARA DETERMINAÇÃO DE $r_{target}$ DO MTD1 . . . . .	70
	APÊNDICE G – CÓDIGO UTILIZADO PARA ESCOLHA DE VA- LORES DE $A$ DO MTD2 . . . . .	71
	APÊNDICE H – FUNÇÃO EM MATLAB PARA OBTENÇÃO DE RESPOSTA ESPERADA DE TREINO DE RNA . . . . .	73
	<b>ANEXOS</b>	<b>74</b>
	ANEXO A – FUNÇÃO EM MATLAB PARA AGRUPAMENTO COM DBSCAN (TRAN; DRAB; DASZYKOWSKI, 2013) . . . . .	75

# 1 Introdução

Sinais de EMG apresentaram crescentes aplicações no controle de próteses mioelétricas. Hargrove et al. (2013) mostraram o controle de uma prótese de perna de um amputado acima do joelho direito, enquanto Chu et al. (2007) apresentaram bons resultados de reconhecimento de padrões de EMG para desenvolvimento de uma prótese multifuncional de mão. Em área paralela ao controle de próteses, Pattichis, Schizas e Middleton (1995) realizaram diagnósticos clínicos de desordens neuromusculares com sinais de EMG e redes neurais artificiais.

As principais estratégias para caracterização de sinais de EMG e potenciais de ação das unidades motoras baseiam-se no uso de um método classificador. Métodos de classificação utilizados incluem - entre inúmeros outros - Redes Neurais Artificiais (HUDGINS; PARKER; SCOTT, 1993), classificador Bayesiano (ANGLEHART; HUDGINS, 2003), lógica *Fuzzy* (CHAN et al., 2000) e *Neuro-Fuzzy* (FAVIEIRO; BALBINOT, 1993). Tais sistemas de classificação necessitam, como parte do preprocessamento, segmentar os sinais de EMG adquiridos, para então realizar extração de características dos segmentos como amplitude, número de cruzamentos por zero, coeficientes de autoregressão, transformadas de Fourier e, mais recentemente, transformadas Wavelet (CHU et al., 2007).

Neste trabalho, é proposto e implementado em MATLAB quatro diferentes métodos de segmentação automática para sinais de EMG de superfície (EMGs). Os primeiros dois métodos (que serão identificados neste estudo pelas abreviações MTD1 e MTD2) tratam da detecção de picos do sinal utilizando *thresholding* e produzem segmentos de comprimento constante centrados nestes picos. O terceiro (MTD3) e quarto (MTD4) métodos utilizam uma janela deslizante para identificação de pontos iniciais e finais dos segmentos, produzindo segmentos de comprimento variável.

O primeiro método (MTD1) é baseado no método de segmentação utilizado em Chauvet et al. (2001). Trata-se de método iterativo, identificando os picos do sinal a partir de *threshold* de amplitude, segmentando o sinal em janelas de comprimento constante centradas nos picos. O valor de *threshold* para a primeira iteração corresponde ao máximo absoluto do sinal. A cada nova iteração em que não se atinge uma razão mínima arbitrada entre número de segmentos e comprimento de sinal, o novo *threshold* é calculado como fração do *threshold* da iteração anterior.

O segundo método (MTD2) é baseado no método de segmentação utilizado em Katsis et al. (2006). De forma similar ao MTD1, também utiliza *threshold* para detecção de picos do sinal e segmentação com janelas de comprimento constante em torno dos picos.

Diferentemente do MTD1, MTD2 não é iterativo, utilizando relação entre valor máximo e valor médio do sinal para cálculo do valor de *threshold*.

O terceiro método (MTD3) é baseado no método de segmentação utilizado em Gut e Moschytz (2000). Uma janela deslizante percorre o sinal, identificando inícios de segmentos quando a variação total no interior da janela excede determinado valor limite. Os finais dos segmentos são identificados quando a variação total do sinal no interior da janela é inferior a um segundo valor de limite.

O quarto método (MTD4) é baseado no método de segmentação utilizado em Pattichis, Schizas e Middleton (1995). Os pontos de início do segmento são identificados quando o valor máximo do sinal contido na janela excede determinado *threshold*. Os respectivos pontos de final de segmento são identificados quando o sinal janelado volta a manter-se abaixo do *threshold*.

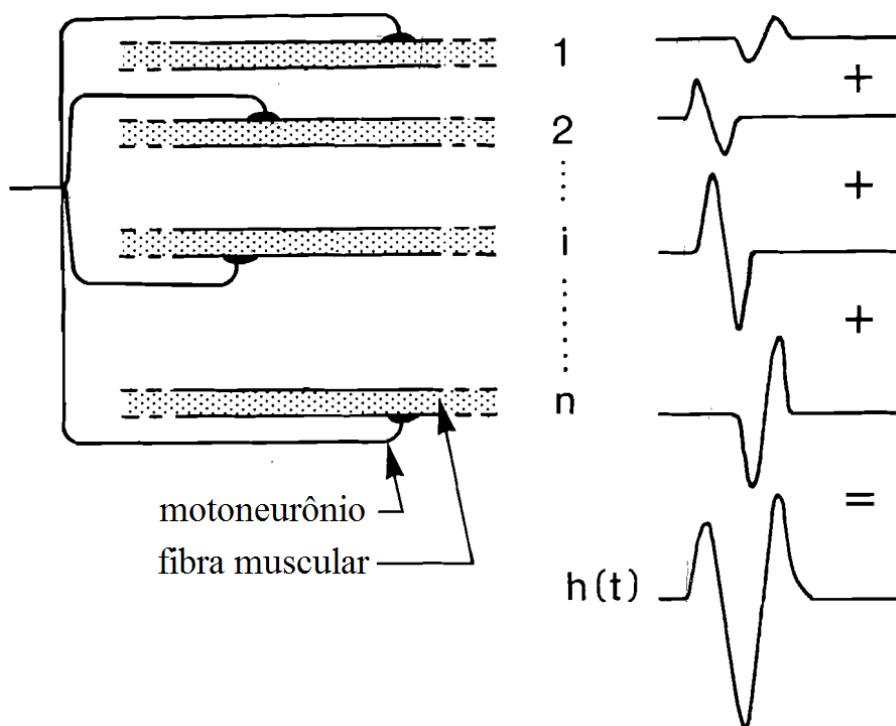
Utilizando valores de RMS, variância e frequência mediana dos segmentos obtidos, Redes Neurais Artificiais (RNAs) foram treinadas para classificar entre 17 classes de movimentos de mão e punho. Os objetivos finais deste trabalho são a implementação dos métodos de segmentação propostos e fornecer avaliação comparativa entre métodos quando utilizados para classificação com uso de RNA.

## 2 Revisão Bibliográfica

### 2.1 Sinais de Eletromiografia

Sinais de EMG podem ser adquiridos por eletrodos posicionados na superfície da pele (eletrodo não invasivo) ou por agulhas introduzidas no tecido muscular (eletrodo invasivo). Os sinais são formados por potenciais de ação de fibras musculares organizadas em unidades funcionais chamadas de “unidades motoras” (MU - *Motor Unit*) (LUCA et al., 2006). Uma MU é composta por um neurônio motor e as fibras musculares que ele inerva, sendo a entidade fundamental que controla a ativação de músculos estriados (BUCHTHAL; SCHMALBRUCH, 1980). A soma algébrica dos potenciais de ação de todas as fibras de uma unidade motora é chamada de “potencial de ação da unidade motora”, ou em inglês, MUAP (*Motor Unit Action Potential*) (ALMEIDA; BARRETO, 1997). A Figura 1 apresenta a composição de uma MUAP a partir da soma dos potenciais das fibras de uma unidade motora.

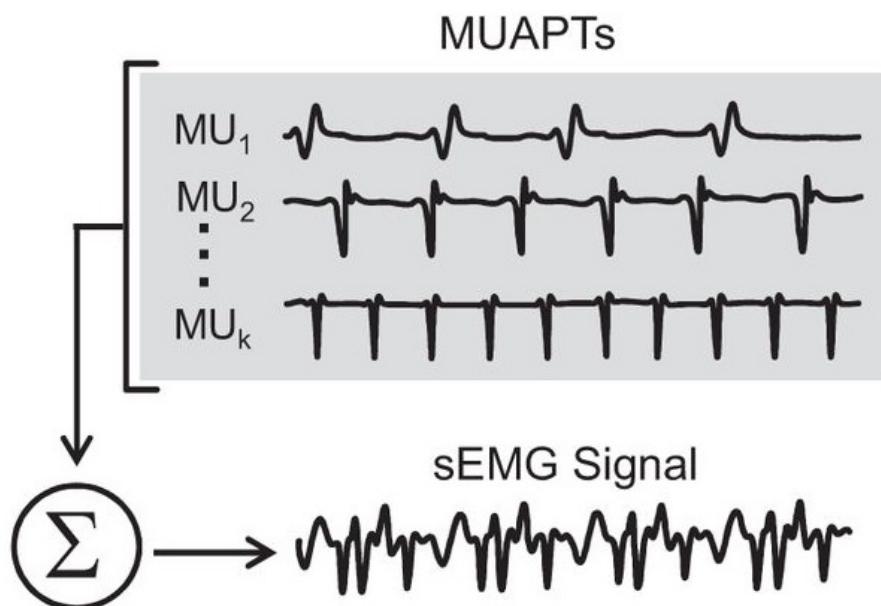
Figura 1 – Soma de potenciais de ação das  $n$  fibras de uma unidade motora, formando uma MUAP  $h(t)$ .



Fonte: adaptado de Basmajian e Luca (1985).

Para os principais métodos utilizados em aquisição do sinal de EMG, é comum a captura da contribuição de mais de uma unidade motora no mesmo canal de aquisição. A influência de uma unidade motora na amplitude do sinal adquirido depende principalmente da distância das fibras musculares ao ponto de aquisição (GERDLE et al., ). Sinais de EMG de longa duração são constituídos por sequências temporais de MUAPs, também conhecidas como MUAPTs (*MUAP Trains*). A Figura 2 exemplifica MUAPTs de diferentes MUs que somam-se para formar um sinal de EMG de longa duração.

Figura 2 – MUAPTs de diferentes MUs somam-se para compor o sinal adquirido por um canal de EMG.



Fonte: adaptado de Kline e Luca (2014)

Para maiores detalhes sobre sinais de eletromiografia de superfície, conceitos sobre anatomia e fisiologia sugere-se a consulta de referências clássicas na área e adicionalmente as seguintes referências: Favieiro (2009), Lopes (2014) e Schons (2014).

## 2.2 Métodos de Segmentação

Esta seção descreve os métodos de segmentação desenvolvidos, citando os trabalhos da área que foram utilizados como base teórica para os métodos.

Nota-se que nomes utilizados para variáveis e constantes (por exemplo, sinal a ser segmentado ' $x$ ', *threshold* ' $T$ ', etc.) foram determinados pelo autor deste trabalho, não necessariamente sendo os mesmos utilizados nos trabalhos citados.

Para as definições dos Métodos 3 e 4 (MTD3 e MTD4) são utilizados os termos BEP (*Beginning Extraction Point*, ponto inicial de um segmento) e EEP (*Ending Extraction Point*, ponto final de um segmento), também utilizados em Pattichis, Schizas e Middleton (1995).

### 2.2.1 Método 1 - método iterativo utilizando *thresholding* para detecção de centros de segmentos de comprimento constante (MTD1)

Este método iterativo é adaptado do método de segmentação utilizado em Chauvet et al. (2001). As definições da Tabela 1 serão utilizados para descrever este método.

Tabela 1 – Parâmetros utilizados para definir o MTD1.

Nome	Descrição
$x$	Sinal a ser segmentado
$L$	Comprimento total do sinal a ser segmentado
$l$	Comprimento desejado para os segmentos
$T_k$	Valor de <i>threshold</i> para a iteração $k$
$T_{lim}$	Valor de limite inferior para o <i>threshold</i>
$q$	Fração de $T_{k-1}$ para determinação de $T_k$
$N_k$	Número total de candidatos para centros de segmentos identificados na iteração $k$
$r_k$	Razão entre número de candidatos identificados na iteração $k$ e o comprimento total do sinal
$r_{target}$	Razão mínima esperada para $r_k$ , utilizada para determinar o final do método

Inicialmente, determina-se o valor de *threshold*  $T_0$  equivalente ao máximo absoluto do sinal a ser segmentado  $x$  (Equação (2.1)). O valor  $T_k$  é atualizado em cada iteração  $k$  como sendo uma fração  $q$  de  $T_{k-1}$  (Equação (2.2)). No trabalho de Chauvet et al. (2001), este valor  $q$  foi empiricamente determinado em 90%.

$$T_0 = \max(x) \quad (2.1)$$

$$T_k = q \times T_{k-1} \quad (2.2)$$

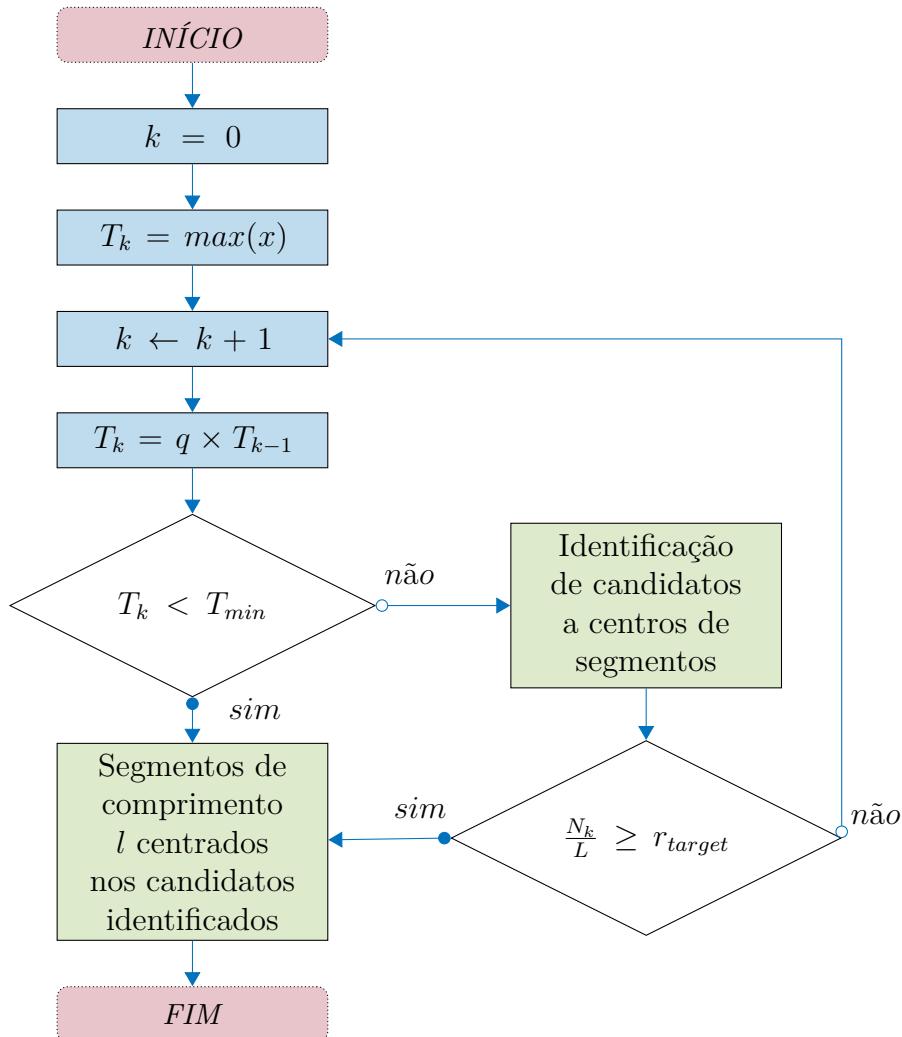
Pontos do sinal acima do valor de  $T_k$  são possíveis candidatos para centros de segmentos. Caso exista mais de um possível candidato em uma vizinhança bilateral de  $l$  amostras do sinal, apenas o ponto de maior amplitude nesta vizinhança é considerado. Para determinar o processo de finalização do método, avalia-se a razão  $r_k$  entre a quantidade identificada de candidatos  $N_k$  e o comprimento total do sinal  $L$  (Equação (2.3)). Caso  $r_k$  seja menor que um valor predeterminado  $r_{target}$ , calcula-se  $T_{k+1}$  para realização da próxima iteração (Equação (2.2)). Caso  $r_k$  seja maior ou igual ao valor predeterminado  $r_{target}$ ,

encerra-se o método e os segmentos são tomados como janelas de sinal de comprimento  $l$ , centradas nos candidatos identificados na última iteração.

$$r_k = \frac{N_k}{L} \quad (2.3)$$

Adicionalmente, o estabelecimento de um valor limite mínimo para  $threshold T_{lim}$  garante que o método não entre em laço infinito e evita detecção de segmentos em trechos de baixa atividade muscular. Caso o valor de  $threshold T_k$  para a iteração atual seja inferior a  $T_{lim}$ , encerra-se o processo iterativo. O método de segmentação MTD1 é representado pelo fluxograma da Figura 3.

Figura 3 – Fluxograma representativo do MTD1.



### 2.2.2 Método 2 - método não iterativo utilizando *thresholding* para detecção de centros de segmentos de comprimento constante (MTD2)

Este é o método de segmentação utilizado por Katsis et al. (2006), que será descrito pelas definições da Tabela 2. Primeiramente, seleciona-se entre dois métodos de cálculo de *threshold*  $T$ : ou utiliza-se  $T$  como múltiplo da média aritmética do sinal  $x$ ; ou  $T$  é uma fração do valor máximo do sinal  $x$ . Katsis et al. (2006) utilizaram a relação do fluxograma da Figura 4 para o cálculo de *threshold*  $T$ .

Tabela 2 – Parâmetros utilizados para definir o MTD2.

Nome	Descrição
$x$	Sinal a ser segmentado
$L$	Comprimento total do sinal a ser segmentado
$l$	Comprimento desejado para os segmentos
$T$	Valor de <i>threshold</i>
$A$	Coeficiente utilizado para decisão de método de cálculo de $T$
$B$	Múltiplo da média aritmética do sinal $x$ para obtenção de $T$
$C$	Fração do valor máximo do sinal $x$ para cálculo de $T$

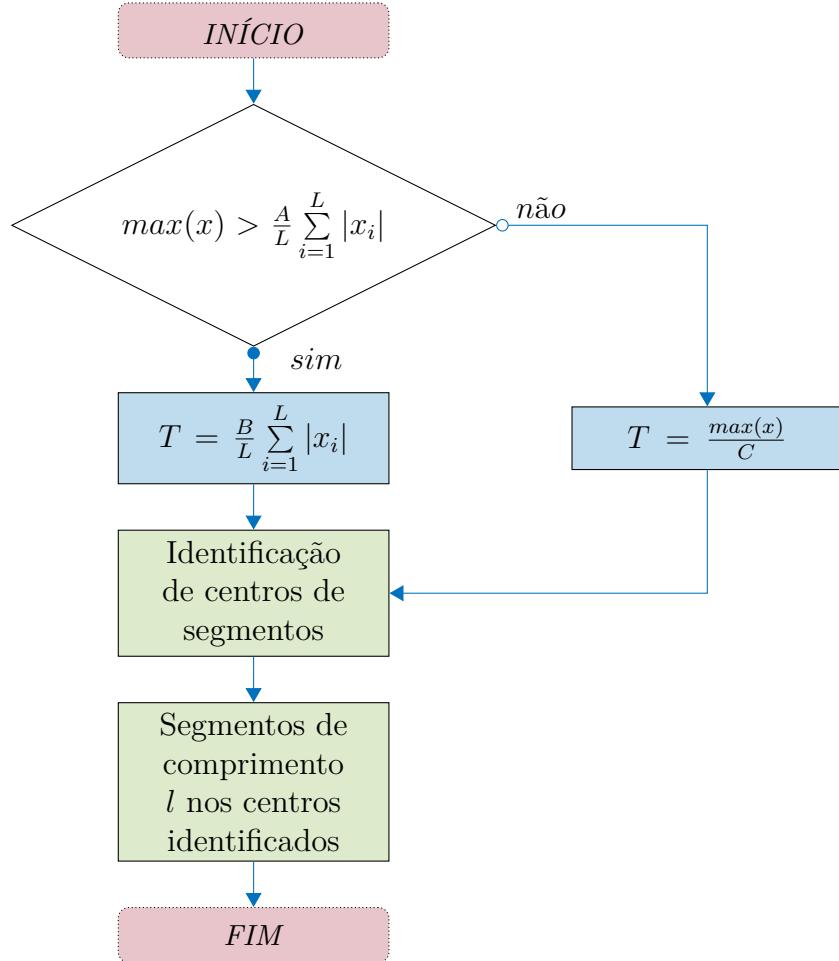
De forma similar ao MTD1, os pontos do sinal que tiverem valor acima de  $T$  são considerados possíveis candidatos para centros de segmentos. Para os possíveis candidatos que estiverem afastados de uma distância inferior a  $l$ , apenas o candidato de maior amplitude é considerado. Em Katsis et al. (2006) foram utilizados coeficientes  $A$ ,  $B$  e  $C$  respectivamente de 30, 5 e 5.

### 2.2.3 Método 3 - método com janela deslizante para detecção de BEP e EEP de segmentos utilizando variação total (MTD3)

Este é o método de segmentação utilizado em Gut e Moschytz (2000). As definições da Tabela 3 serão utilizados para descrever este método. Uma janela deslizante de comprimento  $W$  percorre o sinal da esquerda para a direita. A cada incremento de *step* amostras, caso a variação total  $V$  (Equação (2.4)) do trecho de sinal contido pela janela exceda um limite  $B$  (sendo  $B > 0$ ), o ponto mais à esquerda da janela  $w_0$  determina a BEP de um segmento. O EEP do correspondente segmento é então obtido como o ponto mais à direita ( $w_0 + W$ ) da próxima janela na qual a variação total for menor que um limite  $C$  (sendo  $C < 0$ ). O MTD3 pode ser representado pelo fluxograma da Figura 5.

$$V = \sum_{i=w_0+1}^{w_0+W} (x_i - x_{i-1}) \quad (2.4)$$

Figura 4 – Fluxograma representativo do MTD2.



No método original de Gut e Moschytz (2000), BEPs foram detectadas pelo cálculo da declividade média  $\beta$  no interior da janela (Equação (2.5)) nos pontos que excediam um limite  $B$  (sendo  $B > 0$ ) e EEPs pelo cálculo do módulo da variação total (Equação (2.6)) quando esta era menor que um limite  $C$  (sendo  $C > 0$ ).

$$\beta = \frac{1}{W} \sum_{i=w_0+1}^{w_0+W} (x_i - x_{i-1}) \quad (2.5)$$

$$\gamma = \left| \sum_{i=w_0+1}^{w_0+W} (x_i - x_{i-1}) \right| \quad (2.6)$$

Nota-se que  $\gamma = |W\beta| = |V|$ . O MTD3 proposto, ao utilizar apenas a Equação (2.4) para detecção de BEPs e EEPs, explora tal relação para simplificar o método original, com a vantagem adicional de que os limites  $B$  e  $C$  passam a ser da mesma ordem de grandeza (com o método de Gut e Moschytz (2000),  $B$  seria aproximadamente  $W$  vezes menor que  $C$ ).

Tabela 3 – Parâmetros utilizados para definir o MTD3.

Nome	Descrição
$x$	Sinal a ser segmentado
$l_{\min}$	Distância mínima entre BEPs e EEPs de um mesmo segmento
$l_{\max}$	Distância máxima entre BEPs e EEPs de um mesmo segmento
$W$	Comprimento da janela deslizante utilizada pelo método
$w_0$	Número da amostra mais a esquerda da janela. Determina a posição instantânea da janela
$step$	Número de amostras para incrementar $w_0$ antes de novo cálculo de variação total
$V$	Variação total do sinal $x$ contido na janela deslizante
$\beta$	Declividade média do sinal $x$ contido na janela deslizante
$B$	Valor limite para declividade média que determina um BEP
$\gamma$	Módulo da variação total do sinal $x$ contido na janela deslizante
$C$	Valor limite para variação total que determina um EEP

O incremento de  $step$  amostras (ao invés do avanço de  $w_0$  de uma em uma amostra) serve para reduzir o número de vezes em que é necessário cálculo de variação total  $V$ , simplificando o processamento. Os limites de  $l_{\min}$  e  $l_{\max}$  são necessários para evitar identificação incorreta da EEP relacionada a uma BEP (i.e.  $l_{\max}$  evita que o segmento BEP-EEP contenha sinal respectivo a mais de um movimento e  $l_{\min}$  que o sinal seja segmentado em meio a um movimento).

#### 2.2.4 Método 4 - método com janela deslizante para detecção de BEP e EEP de segmentos utilizando *thresholding* (MTD4)

Este método de segmentação é adaptado de Pattichis, Schizas e Middleton (1995). As definições da Tabela 4 serão utilizados para descrever este método.

Tabela 4 – Parâmetros utilizados para definir o MTD4.

Nome	Descrição
$x$	Sinal a ser segmentado
$l_{\min}$	Distância mínima entre BEPs e EEPs de um mesmo segmento
$l_{\max}$	Distância máxima entre BEPs e EEPs de um mesmo segmento
$W$	Comprimento da janela deslizante utilizada pelo método
$w_0$	Número da amostra mais a esquerda da janela. Determina a posição instantânea da janela
$step$	Número de amostras para incrementar $w_0$ antes de novo cálculo de variação total
$T$	Valor de <i>threshold</i> para o sinal contido na janela

Uma janela deslizante de comprimento  $W$ , com início em  $w_0$ , percorre o sinal da esquerda para a direita. As BEPs de segmentos são os primeiros pontos  $w_0$  em que

o máximo valor de sinal  $x$  contido na janela supera o valor de *threshold*  $T$ . As EEPs associadas às BEPs são as posições  $w_0 + W$  em que tal valor máximo volta a ser inferior a  $T$ . Assim como no MTD3, restrições para comprimento mínimo e máximo ( $l_{\min}$  e  $l_{\max}$ ) de segmentos são utilizadas. O fluxograma da Figura 6 representa o MTD4.

## 2.3 Princípios Básicos sobre Redes Neurais Artificiais

Rede Neural Artificial (RNA) trata-se de um método computacional que, inspirado nos modelos de redes neurais biológicas, “aprende” a resolver determinados problemas (YEGNANARAYANA, 2009), apresentando aplicações em diversos tipos de sinais (e.g. sinais de áudio, imagem e, como é o caso deste trabalho, eletromiografia).

Propostas de utilização de RNA com sinais de EMG para classificação de movimentos datam antes de 1990, a exemplo do trabalho de Hiraiwa, Shimohara e Tokunaga (1989), que menciona a vantagem de RNA sobre outros mecanismos de aprendizado computacional na classificação de padrões que não apresentam separabilidade linear. Desde então, múltiplos trabalhos na área já utilizaram RNA para classificação de movimentos utilizando EMG (e.g. Hudgins, Parker e Scott (1993), Subasi, Yilmaz e Ozcalik (2006), Bu e Fukuda (2003)), apresentando resultados satisfatórios para taxa de acerto de classificação.

### 2.3.1 Modelo Matemático de RNAs

Uma RNA é modelada pelas comunicações de um conjunto de “neurônios” artificiais. Cada neurônio artificial, como mostrado na Figura 7, trata-se de uma função matemática que soma de forma ponderada  $n$  entradas com pesos  $w_n$  e um valor de *bias*  $B$  e utiliza o resultado desta soma em uma função dita “função de ativação” (TANIKIĆ, 2012).

As RNAs utilizadas neste trabalho apresentam arquitetura chamada *feedforward*, onde neurônios são estruturados em camadas consecutivas, de forma que neurônios de uma mesma camada recebem as saídas de neurônios da camada anterior. A Figura 8 exemplifica a estrutura de uma rede neural *feedforward* que utiliza  $m$  entradas para produzir  $n$  saídas, utilizando duas camadas de neurônios chamadas de “camadas ocultas” (i.e. a saída da camada é utilizada como entrada da próxima camada) e uma camada de saída.

Hornik (1991) mostra que ao utilizar neurônios com função de ativação contínua, suave, limitada e não constante, a performance de classificação para uma RNA de arquitetura *feedforward* é arbitrariamente boa a partir, desde que um número suficiente de neurônios artificiais sejam utilizados nas camadas ocultas. Em aplicações de reconhecimento de padrões, normalmente utiliza-se uma função de ativação do tipo sigmoide (i.e. função cuja curva lembra a forma de um “S”). A função de ativação utilizada para RNAs, neste trabalho, é a função sigmoide logística *logsig()*, dada pela Equação (2.7).

$$\text{logsig}(x) = \frac{1}{1 + e^{-x}} \quad (2.7)$$

### 2.3.2 Treinamento de RNAs

Uma RNA pode ser utilizada para resolver determinado problema após um processo de treinamento supervisionado. Conjuntos de dados de treinamento (entradas e respostas esperadas) são apresentados à RNA. Um algoritmo iterativo modifica os pesos e *bias* dos neurônios buscando combinação que otimize o desempenho da RNA para os dados de treinamento. Às iterações do algoritmo de treinamento, em que os dados de treinamento são rerepresentados à RNA, dá-se o nome de *epoch*. O algoritmo utilizado para treinamento de RNAs, neste trabalho, chama-se *scaled conjugate gradient backpropagation* (MØLLER, 1993) (identificado em MATLAB como *trainscg*), que apresenta boa performance em problemas de reconhecimento de padrões.

Quando uma RNA é aplicada para classificação de séries temporais de sinais, as entradas (chamadas “preditores”) utilizadas pela RNA são características extraídas de um trecho do sinal (e.g. número de cruzamentos por zero, valor médio, valor RMS). A extração de características tem o objetivo de reduzir a dimensionalidade do vetor de entradas da RNA (i.e. ao invés de um trecho de sinal com grande número de amostras, a entrada da RNA trata-se de um número arbitrariamente pequeno de preditores), o que apresenta vantagens em desempenho computacional e contorna problemas que surgem da classificação de sinais ruidosos (KIM; HA, 2000).

Para maiores detalhes sobre RNAs e outros métodos de aprendizado computacional, sugere-se consulta das seguintes referências: YEGNANARAYANA (2009), Abraham (2005) e Drew e Monson (2000).

### 2.3.3 Valor $F$

Em análises de sistemas classificadores, costumeiramente utiliza-se uma tabela conhecida como “matriz de confusão”. A Tabela 5 exemplifica uma matriz de confusão de um classificador dito “binário”, de forma que resultados de classificação possíveis pertencem a duas classes - “positiva” (também conhecida como “condição presente”) ou “negativa” (“condição ausente”).

Extrapolar a Tabela 5 para o caso das RNAs deste trabalho, que realizam classificação entre 17 classes de movimentos, significa obter tabela equivalente para cada uma das classes de movimento  $\zeta$  e respectivos valores  $a_\zeta$ ,  $b_\zeta$ ,  $c_\zeta$  e  $d_\zeta$ , interpretando “Positiva” como a identificação de ocorrência do movimento de determinada classe e “Negativa” a não-ocorrência do mesmo.

Tabela 5 – Exemplo de matriz de confusão.

		Classificação obtida	
		Negativa	Positiva
Classificação esperada	Negativa	<i>a</i>	<i>b</i>
	Positiva	<i>c</i>	<i>d</i>

*a*: verdadeiros negativos

*b*: falsos positivos

*c*: falsos negativos

*d*: verdadeiros positivos

Fonte: adaptado de Kubat, Holte e Matwin (1998).

Em situações onde existe grande disparidade entre o número de classificações esperadas positivas e negativas, é comum o uso de indicadores de precisão  $p_\zeta$  (Equação (2.8)) e sensitividade  $r_\zeta$  (Equação (2.9)) por classe  $\zeta$  - que não utilizam em suas definições o número geralmente grande de classificações verdadeiras negativas  $a_\zeta$  - e combinações destes indicadores (KUBAT; HOLTE; MATWIN, 1998). O valor  $F_{\zeta_\nu}$  para determinada classe  $\zeta$  e volutário  $\nu$  é definido como a média harmônica entre precisão e sensitividade, conforme a Equação (2.10).

$$p_\zeta = \frac{d_\zeta}{b_\zeta + d_\zeta} \quad (2.8)$$

$$r_\zeta = \frac{d_\zeta}{c_\zeta + d_\zeta} \quad (2.9)$$

$$F_{\zeta_\nu} = 2 \times \frac{p_{\zeta_\nu} \times r_{\zeta_\nu}}{p_{\zeta_\nu} + r_{\zeta_\nu}} \quad (2.10)$$

Para avaliação de desempenho de classificação de RNAs, este trabalho utilizará valor  $F$  médio  $\bar{F}_\zeta$  por classe de movimento  $\zeta$  entre os  $M$  voluntários de uma mesma base de dados (definido pela Equação (2.11)) e valor  $F$  médio entre as 17 classes de movimento e  $M$  voluntários  $\bar{\bar{F}}$  (Equação (2.12)).

$$\bar{F}_\zeta = \frac{1}{M} \sum_{\nu=1}^M F_{\zeta_\nu} \quad (2.11)$$

$$\bar{\bar{F}} = \frac{1}{17} \sum_{\zeta=1}^{17} \bar{F}_\zeta \quad (2.12)$$

Figura 5 – Fluxograma representativo do MTD3.

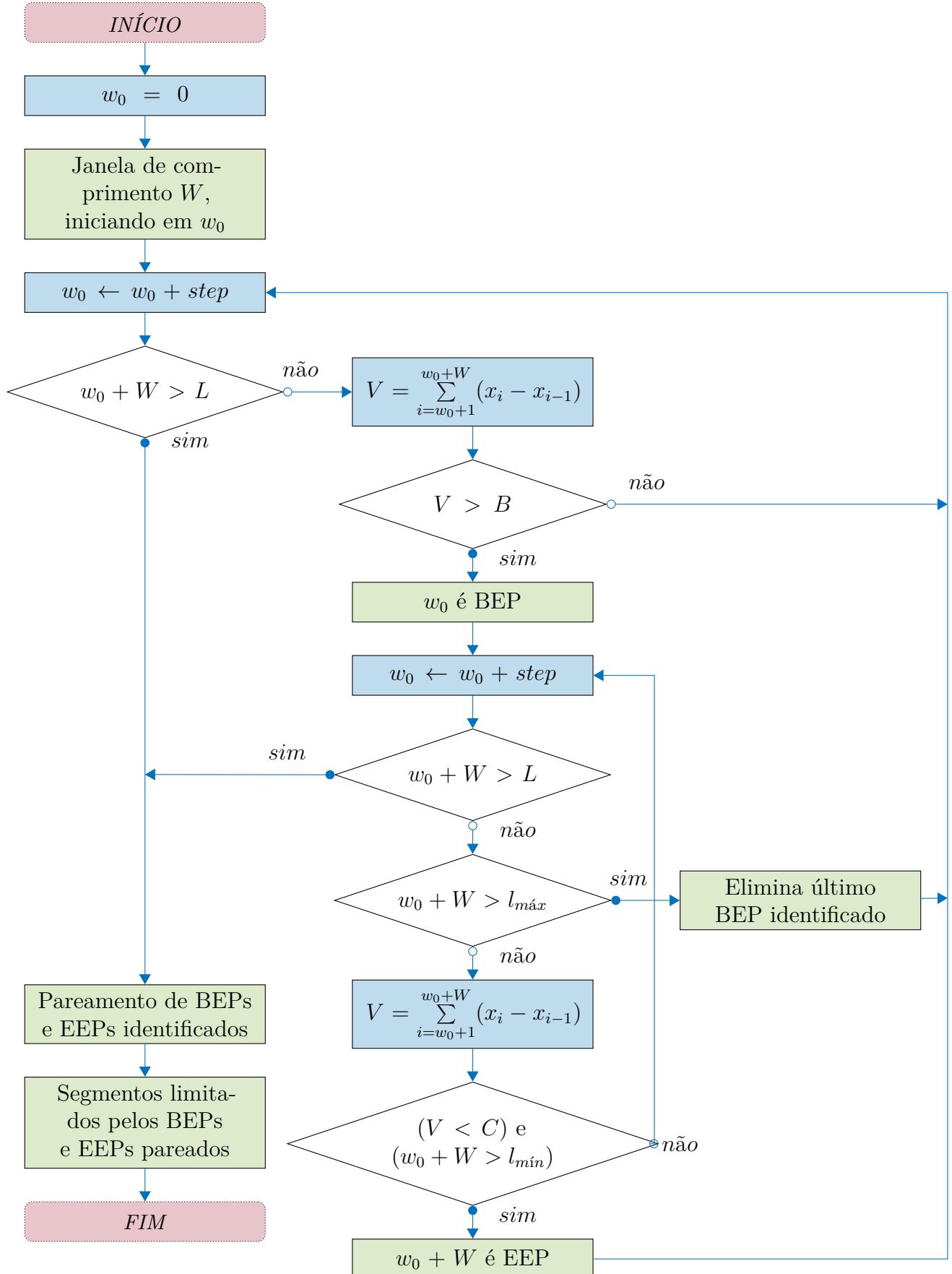


Figura 6 – Fluxograma representativo do MTD4.

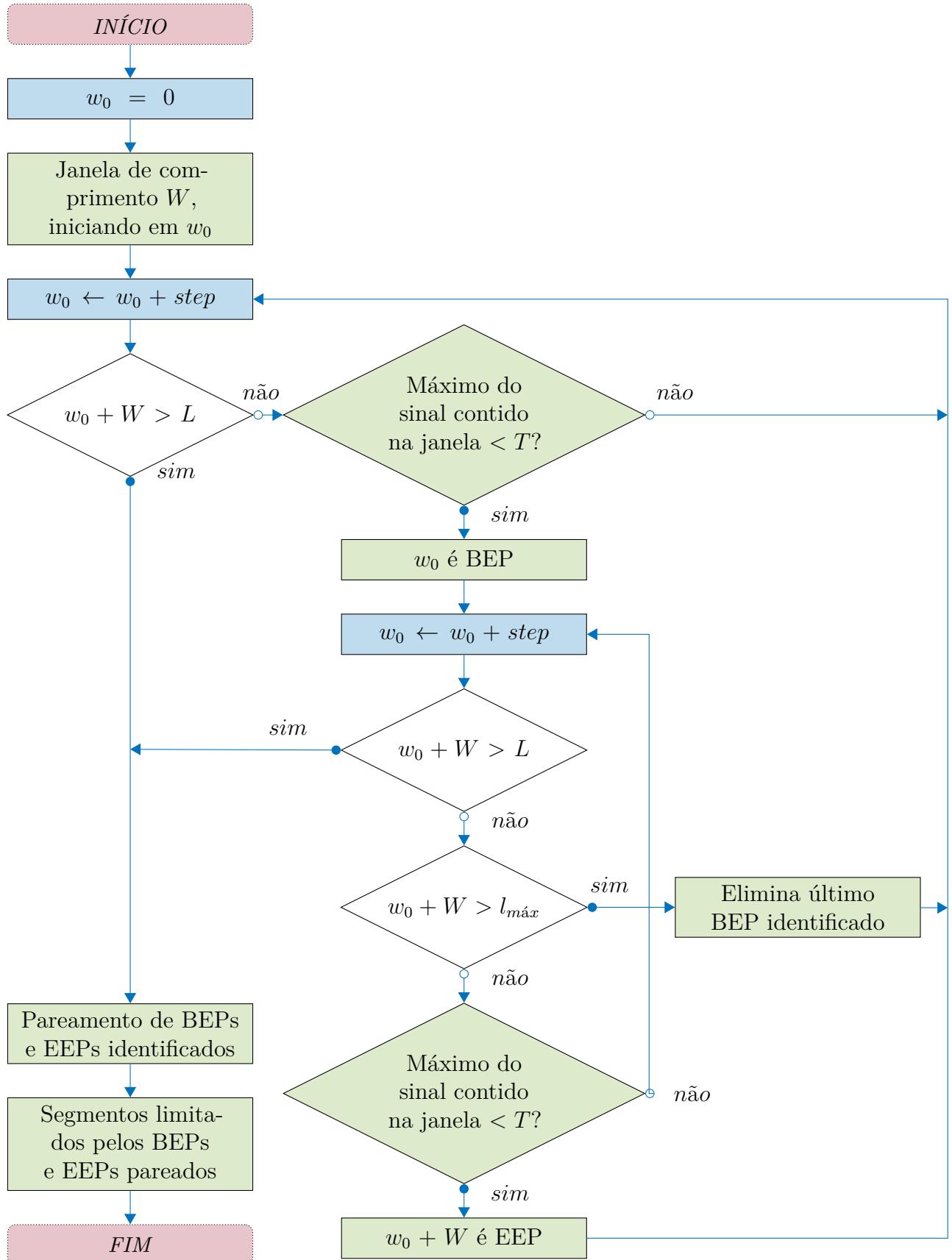
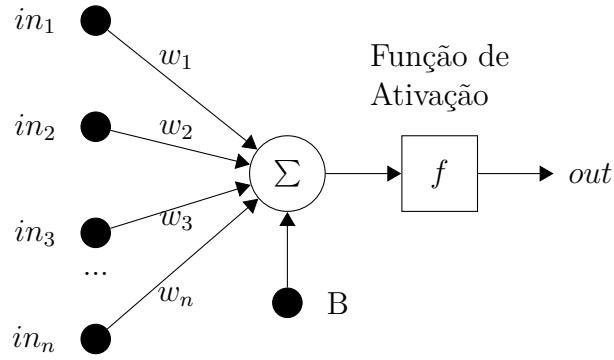
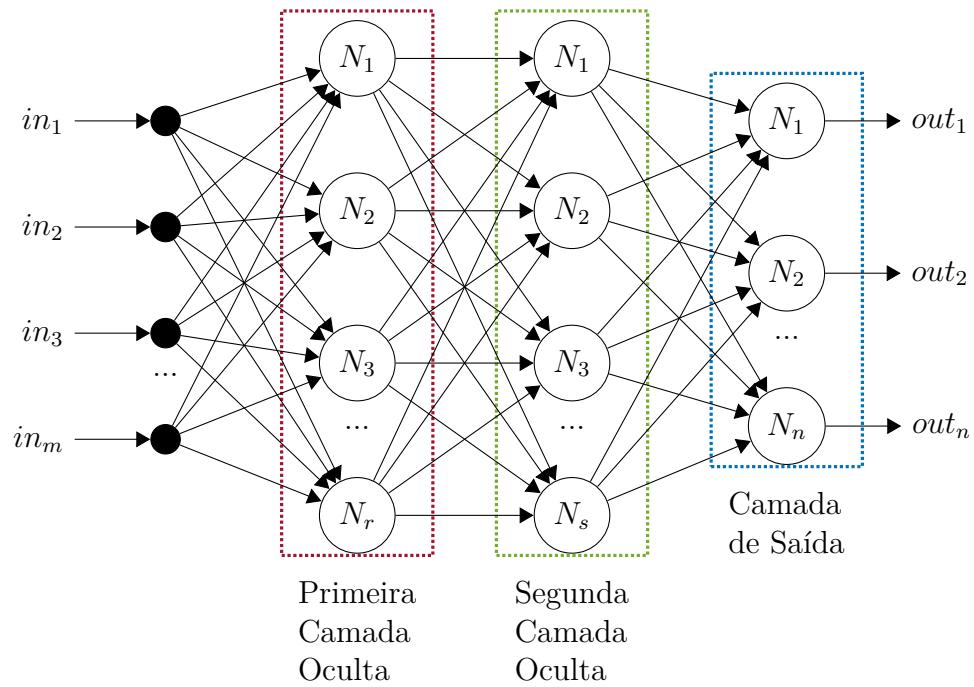


Figura 7 – Diagrama representativo do modelo de um neurônio artificial genérico.



Fonte: adaptado de Tanikić (2012)

Figura 8 – Exemplo de uma RNA em arquitetura *feedforward*.



Fonte: adaptado de Barbosa, Freitas e Neves (2005)

# 3 Metodologia Experimental

## 3.1 Bases de Dados Utilizadas

Este trabalho faz uso de aquisições para o exercício 1 da base de dados número 2 do projeto NinaPro (GIJSBERTS et al., 2014) e a base de dados em construção realizada pelo Laboratório de Instrumentação Eletro-Eletrônica (IEE).

A base de dados do IEE busca replicar os métodos de aquisição utilizados pelo projeto NinaPro, contando com as mesmas definições de movimentos realizados e posicionamento de eletrodos e mesmo período de amostragem ( $500 \mu s$ ). A base de dados NinaPro é composta por sinais de aquisição com 40 voluntários, enquanto a base de dados do IEE, ainda em construção, contém atualmente sinais de 10 voluntários.

### 3.1.1 Posicionamento de Eletrodos

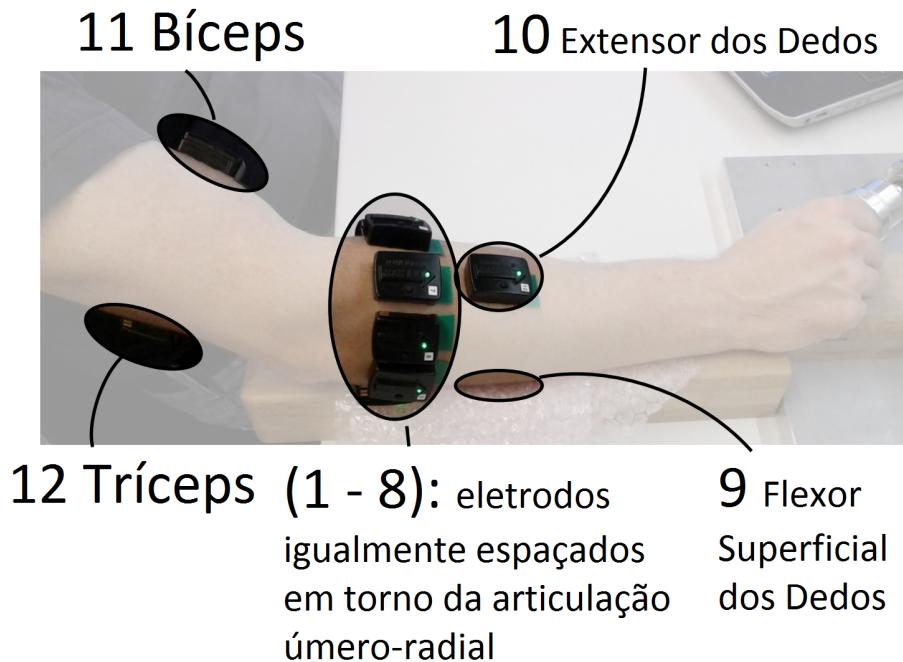
Os sinais de EMG de superfície para ambas as bases utilizadas neste trabalho são compostos por canais de aquisição de doze eletrodos posicionados no braço direito de voluntários saudáveis (i.e. não amputados e sem desordens neuromusculares). Os primeiros oito canais correspondem a eletrodos posicionados de forma a circundar o antebraço e a junção úmero-radial. O eletrodo do canal número 9 é posicionado sobre o músculo flexor superficial dos dedos e o eletrodo do canal 10, em oposição, é posicionado sobre o músculo extensor dos dedos. Os últimos dois eletrodos, 11 e 12, são posicionados sobre o bíceps e o tríceps, respectivamente. A Figura 9 apresenta o posicionamento para os doze eletrodos de superfície e a numeração de seus respectivos canais que compõem os sinais para ambas as bases de dados.

### 3.1.2 Sistema para Aquisição de EMGs no IEE

Para aquisições realizadas no IEE, os eletrodos são conectados a dois sistemas de aquisição EMG 830C, produzidos pela empresa “EMG *System* do Brasil”, de oito canais cada (os canais de número 1 a 8, que circundam a junção úmero-radial, são conectados ao primeiro sistema e os canais 9 a 12 ao segundo sistema). Os sinais de referência para os canais são tomados de eletrodos posicionados na testa do voluntário.

Os sistemas EMG 830C são conectados a um computador pelo conjunto de *Data Acquisition* (DAQ) NI SCB-68A e NI USB-6289. O computador executa uma rotina de aquisição de sinais implementada em NI LabView que é também responsável por exibir

Figura 9 – Posicionamento de eletrodos de superfície no braço de um voluntário.



Fonte: adaptado de Gijsberts et al. (2014)

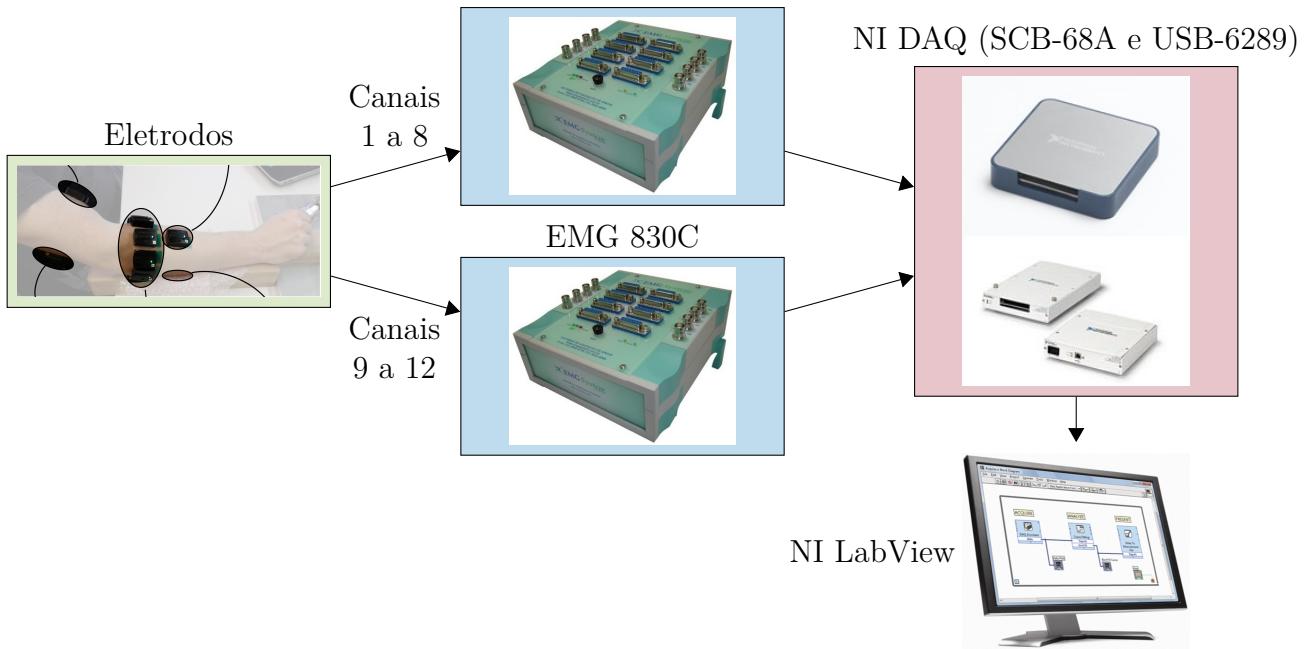
um vídeo com movimentos a serem replicados pelo voluntário. O diagrama da Figura 10 resume o sistema de aquisição para a base de dados do IEE.

### 3.1.3 Movimentos de Interesse

Após o posicionamento de eletrodos, os voluntários sentam-se em frente a um monitor e apoiam o braço direito de forma relaxada sobre uma superfície horizontal próxima à altura do cotovelo. O monitor exibe um vídeo com uma mão direita, digitalmente animada, realizando movimentos de interesse que devem ser replicados pelos voluntários. A Figura 11 apresenta a mão digitalmente animada realizando alguns dos movimentos de interesse deste trabalho.

Primeiramente, realiza-se um “treinamento” com o voluntário, exibindo o vídeo e apresentando os movimentos que devem ser realizados (sem aquisição de sinal), de modo a reduzir possíveis erros na realização de movimentos para uma segunda exibição, quando os sinais de EMG serão devidamente adquiridos. Cada movimento apresentado em vídeo tem duração de 5 segundos, com um intervalo de 3 segundos de repouso entre movimentos. O voluntário realiza 6 repetições consecutivas para os 17 movimentos descritos na Tabela 6.

Figura 10 – Diagrama de blocos para o sistema de aquisição da base de dados do IEE.



Fonte: montagem com imagens de Gijsberts et al. (2014) e dos *websites* das empresas *EMG Systems* do Brasil e *National Instruments*

## 3.2 Métodos de Segmentação

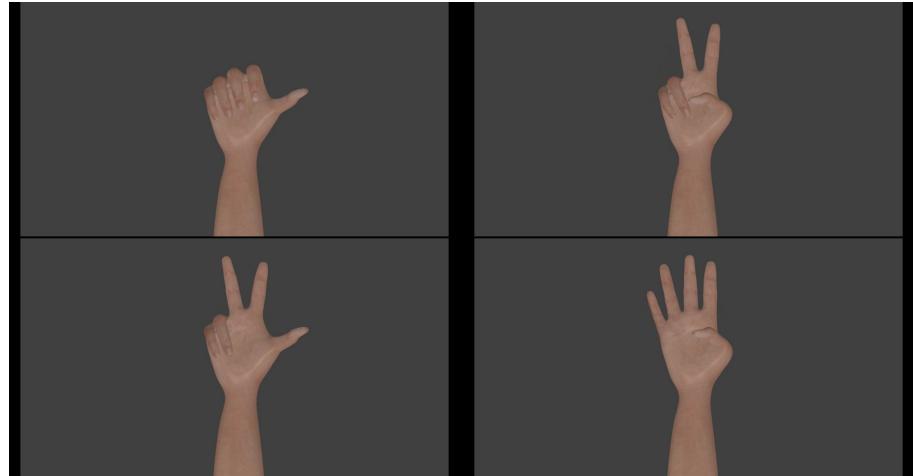
Esta seção descreve a implementação dos métodos de segmentação propostos. O diagrama de blocos da Figura 12 apresenta de forma resumida os passos comuns aos quatro métodos, que serão explanados nas subseções seguintes. Os códigos criados para os quatro métodos, escritos para MATLAB R2015a, encontram-se nos Apêndices A a D.

### 3.2.1 Preprocessamento

Os sinais de eletromiografia para ambas as bases de dados (NinaPro e IEE) são armazenados mantendo sua polaridade original (i.e. amostras do sinal podem assumir valores positivos e negativos). Primeiramente, realiza-se a retificação completa dos sinais tomando o módulo dos valores amostrados (em MATLAB, função *abs()*). A retificação completa do sinal mantém sua energia e é fundamental para a implementação dos métodos de segmentação aqui desenvolvidos.

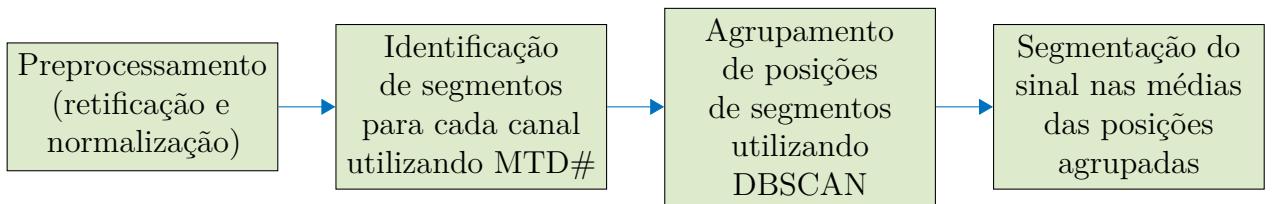
Após a etapa de retificação, os sinais para cada canal de aquisição são normalizados de acordo com seu valor máximo, de modo que seu novo valor máximo seja unitário, a partir da Equação (3.1), onde  $x_j$  é o sinal original para um canal  $j$  e  $x_{j,norm}$  é sua versão normalizada. A normalização de canais faz com que os parâmetros utilizados pelos métodos de segmentação sejam relativos ao valor máximo do sinal, possibilitando a implementação

Figura 11 – Cenas do vídeo apresentado aos voluntários na aquisição do IEE.



Fonte: Laboratório de Instrumentação Eletro-Eletrônica, UFRGS

Figura 12 – Diagrama de blocos geral para os métodos de segmentação.



para diferentes voluntários. A Figura 13 exemplifica retificação e normalização para trecho de sinal de um canal de EMG.

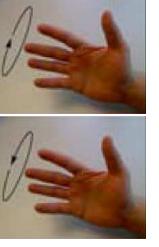
$$x_{j_{norm}} = \frac{x_j}{\max(x_j)} \quad (3.1)$$

### 3.2.2 Parâmetros Ajustáveis

Cada método de segmentação MTD1 - MTD4 apresenta um conjunto de parâmetros ajustáveis. Tais parâmetros foram descritos anteriormente na Seção 2.2 (Tabelas 1 a 4). Após investigações iniciais de segmentações obtidas com diferentes valores de parâmetros, listou-se valores a serem explorados na aplicação de cada método. A Tabela 7 apresenta os parâmetros ajustáveis de cada método e sua respectiva lista de valores explorados.

Para os métodos que necessitam parâmetro de comprimento de segmento  $l$  (i.e. os métodos que produzem segmentos de comprimento de janela constante, MTD1 e MTD2) utilizou-se valor de  $l$  como  $10 \times 10^3$ , que corresponde a 5 cinco segundos de aquisição

Tabela 6 – Lista de movimentos de interesse apresentados aos voluntários.

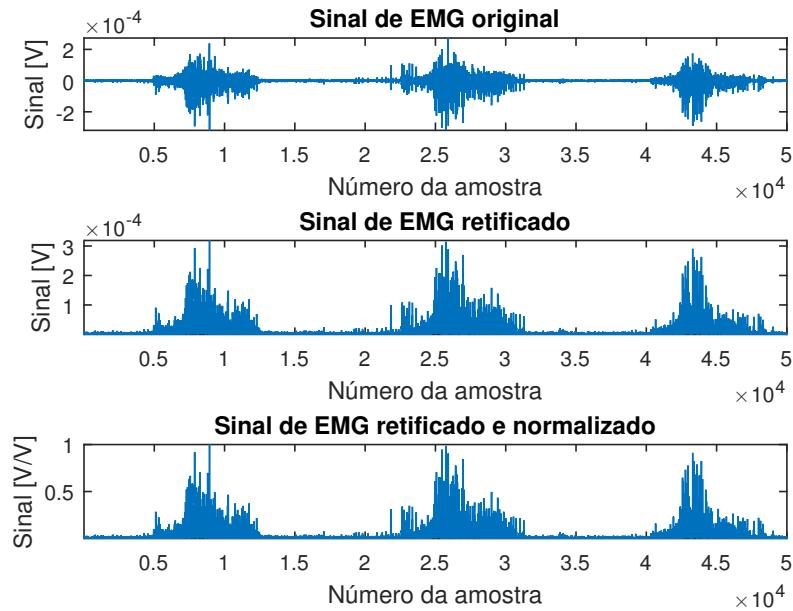
#	Descrição	Imagen demonstrativa	#	Descrição	Imagen demonstrativa	#	Descrição	Imagen demonstrativa
1	Polegar esticado, flexão dos outros dedos.		2	Extensão do indicador e dedo médio, flexão dos outros dedos.		3	Flexão do dedo anelar e mínimo, extensão dos outros dedos.	
4	Polegar para a base do dedo mínimo.		5	Abdução (“afastamento”) de todos os dedos extendidos.		6	Flexão de todos os dedos ao punho.	
7	Extensão do indicador em movimento de “apontar”.		8	Adução (“aproximação”) de todos os dedos extendidos.		9 10	Rotação de punho em torno do dedo médio, dois sentidos.	
11 12	Rotação de punho em torno do dedo mínimo, dois sentidos.		13	Flexão de punho.		14	Extensão de punho.	
15	Desvio radial do punho.		16	Desvio ulnar do punho.		17	Extensão de punho com mão cerrada.	

Fonte: adaptado de Atzori et al. (2014).

para ambas as bases de dados (período de amostragem para ambas é de  $500 \mu s$ ), sendo a mesma duração dos segmentos de vídeo replicados pelos voluntários.

Para os métodos que utilizam janela deslizante (MTD3 e MTD4), utilizou-se comprimento de janela  $W$  de 5000 amostras, de modo que o comprimento temporal da janela (2,5 segundos) equivale à metade da duração do segmento de vídeo, com incrementos *step* de 100 amostras. Os parâmetros de  $l_{\min}$  e  $l_{\max}$  utilizados foram de 7500 e 12500 amostras, respectivamente.

Figura 13 – Exemplo para retificação e normalização de trecho de sinal de EMG.



O valor de  $r_{target}$  utilizado em cada base de dados no MTD1 foi obtido utilizando o código do Apêndice F como sendo valor mínimo da razão entre número de segmentos e comprimento de sinal para os voluntários de cada base de dados da NinaPro e IEE.

Os valores de  $A$  para o MTD2 foram escolhidos utilizando o código do Apêndice G de modo a estarem em torno da mediana para razão entre valor máximo e valor médio de sinais. Desta forma, assegura-se que o cálculo de *threshold* deste método apresente situações numericamente representativas para uso de ambos parâmetros  $B$  e  $C$ .

### 3.2.3 Agrupamento das Posições de Segmentos de Diferentes Canais com DBSCAN

Como já mencionado, os sinais para ambas as bases de dados são compostos por doze canais de aquisição. Os métodos de segmentação são implementados individualmente para os doze canais. Para os métodos MTD1 e MTD2, posições centrais dos segmentos obtidas em cada canal são armazenadas, enquanto que para os métodos MTD3 e MTD4 armazena-se as posições de BEPs e EEPs. Tais posições de diferentes canais são identificadas como pertencendo a um mesmo segmento através do algoritmo de agrupamento DBSCAN (*Density-Based Spatial Clustering of Applications with Noise*).

DBSCAN foi inicialmente proposto em Ester et al. (1996) para agrupamento de pontos utilizando dois parâmetros:  $\epsilon$ , que define uma vizinhança em torno do valor de cada ponto, e  $minPts$ , que determina o número mínimo de pontos pertencentes à mesma

Tabela 7 – Parâmetros ajustáveis para os métodos de segmentação.

Método	Parâmetros	Valores utilizados	Número total de combinações
MTD1	$l$	$10 \times 10^3$	
	$r_{target}$	NinaPro: $5,6 \times 10^{-5}$ ; IEE: $6,1 \times 10^{-5}$	
	$q$	[0, 75 0, 80 0, 85 0, 90 0, 95]	
	$T_{lim}$	[0, 05 0, 10 0, 15 0, 20 0, 75]	25
MTD2	$l$	$10 \times 10^3$	
	$A$	[60 80 100]	
	$B$	[2 5 8]	
	$C$	[2 5 8]	27
MTD3	$W$	$5 \times 10^3$	
	$step$	100	
	$l_{\min}$	$7,5 \times 10^3$	
	$l_{\max}$	$12,5 \times 10^3$	
	$B$	[0,05 0,10 0,15 0,20 0,25]	25
MTD4	$W$	$5 \times 10^3$	
	$step$	100	
	$l_{\min}$	$7,5 \times 10^3$	
	$l_{\max}$	$12,5 \times 10^3$	
	$T$	[0,01 0,02 0,03 ... 0,24 0,25]	25

vizinhança para formação do “núcleo” de um grupo. Para mais detalhes sobre DBSCAN, sugere-se o artigo original (ESTER et al., 1996) e a revisão do método (TRAN; DRAB; DASZYKOWSKI, 2013), cujo código em MATLAB foi utilizado (Anexo A).

Na implementação dos quatro métodos, utilizou-se  $\epsilon = 2000$  amostras e  $minPts = 3$  canais. Em outras palavras, isto significa que para a identificação de um centro de segmento (métodos MTD1 e MTD2) ou de BEPs e EEPs (métodos MTD3 e MTD4) é necessário que as posições identificadas, em pelo menos 3 diferentes canais, pertençam a uma vizinhança bilateral de 4000 amostras (2 segundos de aquisição).

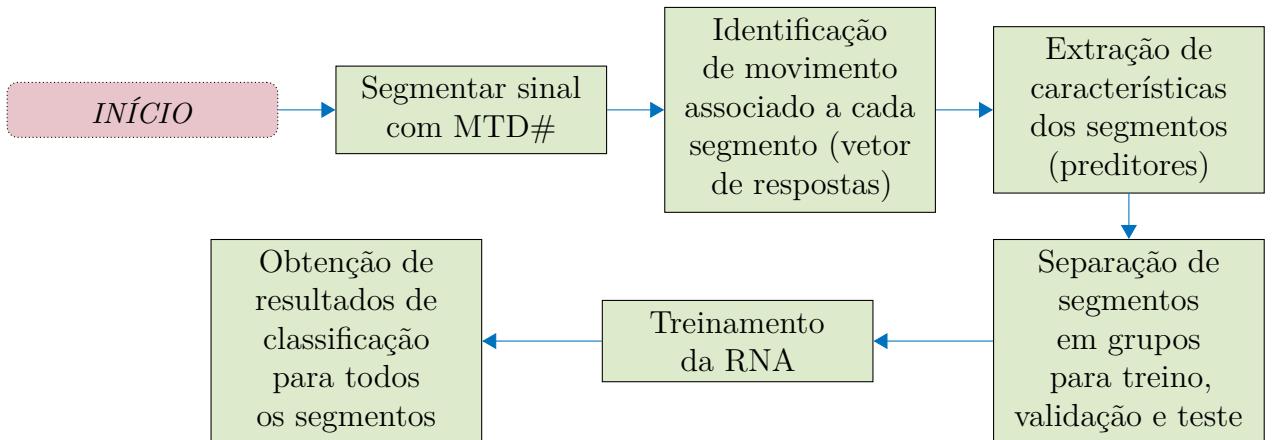
O algoritmo DBSCAN foi escolhido para esta aplicação por dois motivos principais. Primeiramente, ao contrário de métodos de agrupamento como *k-means*, DBSCAN não necessita parâmetro que especifique número de grupos, no caso, número de segmentos a serem obtidos, o que mantém o caráter “automático” dos métodos de segmentação. Além disto, o parâmetro  $minPts$  faz com que este método seja capaz de descartar posições *outliers* que venham a ser identificadas por canal. Sendo assim, o algoritmo DBSCAN é capaz de tornar mais robusto os métodos de segmentação e implementáveis a sinais de múltiplos canais.

Após o agrupamento de posições de segmentos obtidos nos diferentes canais, tomam-se as médias de cada grupo como as posições nas quais o sinal deve ser segmentado (para os métodos MTD1 e MTD2, as médias dos grupos indicam posições centrais de segmentos e para métodos MTD3 e MTD4 indicam posições de BEPs e EEPs). O sinal é segmentado nestas posições para os doze canais, mantendo a coerência temporal entre canais.

### 3.3 Classificação de Segmentos com Redes Neurais Artificiais

Esta seção descreve o uso de RNAs para classificação dos segmentos obtidos de acordo com movimentos de interesse. O processo de classificação é representado pelo diagrama de blocos da Figura 14, que será explanado nas subseções seguintes.

Figura 14 – Diagrama de blocos para processo de classificação de movimentos.



#### 3.3.1 Características Utilizadas como Preditores

Os preditores (“entradas”) das RNAs são o valor RMS, a variância e a frequência mediana do espectro de potência dos segmentos de sinal obtidos pelos métodos MTD1 - MTD4. Tais características foram selecionados de acordo com outros trabalhos já realizados no Laboratório de Instrumentação Eletro-Eletrônica que obtiveram bons resultados em métodos de inteligência computacional, como Favieiro (2009), Schons (2014) e Cene e Balbinot (2015).

O valor RMS ou *Root Mean Square* de um sinal discreto  $x$  de comprimento  $L$  (em MATLAB, função `rms()`) é dada pela Equação (3.2) e a variância deste sinal (em MATLAB, função `var()`) é dada pela Equação (3.3), onde  $\bar{x}$  é o valor médio do sinal. A frequência mediana é tal que a soma total da densidade de potência para frequências abaixo da frequência mediana é igual à soma total da densidade de potência para frequências acima da frequência mediana, sendo estimada pela função `medfreq()` em MATLAB.

$$rms(x) = \sqrt{\frac{1}{L} \sum_{i=1}^L x_i^2} \quad (3.2)$$

$$var(x) = \frac{1}{L-1} \sum_{i=1}^L (x_i - \bar{x})^2 \quad (3.3)$$

### 3.3.2 Classes de Movimentos Utilizadas como Resposta

Realiza-se a identificação de movimentos associados a cada segmento obtido a partir de sua posição temporal no sinal original, já que em ambas as bases de dados os movimentos são realizados na mesma sequência para todos os voluntários.

A base de dados NinaPro contém um *array stimulus* que indica os instantes em que foi exibida em vídeo cada classe de movimentos a serem replicados pelo voluntário. Este *array* é utilizado para obter a relação entre posições de segmentos e classes de movimentos.

A base de dados IEE não contém tal *array*. A classe de movimento associada ao segmento é identificada a partir da posição do segmento relativa ao comprimento total do sinal - considera-se que cada décima-sétima parte do comprimento total do sinal corresponde a uma classe de movimentos distinta.

A função apresentada no Apêndice H é utilizada para criar o vetor de respostas esperadas no treinamento da RNA, tendo como argumentos as posições centrais de segmentos (para os métodos MTD3 e MTD4, calcula-se média dos pares BEP-EEP), uma *string* identificadora da base de dados utilizada e, dependendo da base utilizada, *array stimulus* ou o comprimento total do sinal  $L$ .

### 3.3.3 Separação de Grupos de Treino, Validação e Teste para Treinamento

Sendo  $N_\zeta$  o número de segmentos obtidos por classe de movimentos  $\zeta$ , os grupos de segmentos utilizados nos treinamentos de RNAs são tais que:

- grupo de treino: primeiros  $N_\zeta - 2$  segmentos obtidos de todas as classes de movimentos
- grupo de validação: segmento obtido de índice  $N_\zeta - 1$  de todas as classes de movimentos
- grupo de teste:  $N_\zeta$ -ésimo segmento obtido de todas as classes de movimentos
- Para  $N_\zeta = 2$ , o primeiro segmento identificado da classe  $\zeta$  entra no grupo de treino e o segundo segmento no grupo de validação.
- Para  $N_\zeta = 1$ , o segmento identificado da classe  $\zeta$  entra no grupo de treino.

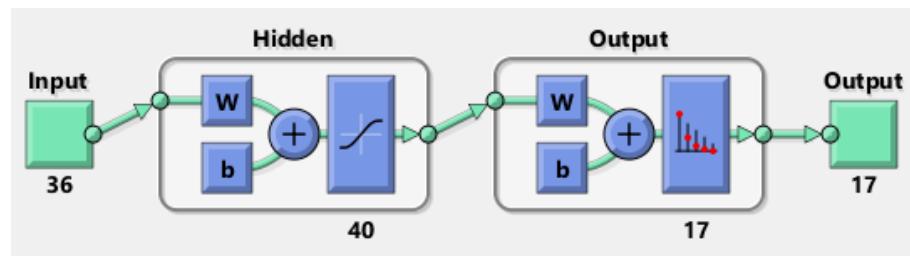
Optou-se por este controle da divisão por indexação (método de divisão chamado em MATLAB de *divideind*) ao invés de métodos de divisão aleatória de grupos (*dividerand*) devido ao número reduzido de repetições por classe de movimento (6 repetições). Utilizar divisão aleatória de grupos apresentaria alta chance de situações particulares em que nenhum (ou, na situação oposta e igualmente indesejada, todos) segmentos de determinada classe estariam inclusos no grupo “treino”, viciando resultados de classificação.

### 3.3.4 Estrutura de RNAs Treinadas

As RNAs utilizadas para todos os métodos são do tipo *feedforward*, compostas por uma camada oculta e uma camada de saída, sendo a camada oculta formada por 40 neurônios com função de ativação *logsig()* (Equação (2.7)) e a camada de saída por 17 neurônios, um para cada classe de movimento, sendo que nesta camada o neurônio com saída de maior valor indica qual a classe de movimento em que foi classificado o segmento.

A Figura 15 mostra a representação esquemática realizada pelo MATLAB para a estrutura das RNAs. O bloco *Input* (“entrada”) indica o uso total de 36 preditores, referentes a segmentos de sinal com 12 canais e 3 preditores por canal (RMS, variância e frequência mediana). Na camada *Hidden* (“oculta”) indica-se o uso de 40 neurônios com função de ativação *logsig()* e a camada *Output* (“saída”) representa os 17 neurônios que indicam resultados de classificação.

Figura 15 – Estrutura de RNAs utilizadas.



# 4 Resultados e Discussões

## 4.1 Segmentação e Classificação por Combinações de Parâmetros

As diferentes combinações de parâmetros testadas nos métodos MTD1 a MTD4 foram indexadas por método de segmentação de acordo com a Tabela 8. As Figuras 16 a 19 apresentam resultados de média e moda de número de segmentos obtidos por classe de movimento com a implementação de cada combinação de parâmetros dos métodos a todos os voluntários de cada base de dados. Lembra-se que no método experimental cada voluntário realiza seis repetições por classe de movimento. Também são apresentados resultados para valor  $F$  médio entre todas as classes de movimentos e voluntários de uma mesma base de dados,  $\bar{F}$  (Equação (2.12)).

Tabela 8 – Combinações de parâmetros utilizados nos métodos de segmentação.

Método#	1		2			3		4	
Parâmetro	q	$T_{lim}$	A	B	C	B	C	T	
Índice da Combinacão	1	0,75	0,05	60	2	2	0,05	-0,05	0,01
	2	0,80	0,05	80	2	2	0,10	-0,05	0,02
	3	0,85	0,05	100	2	2	0,15	-0,05	0,03
	4	0,90	0,05	60	5	2	0,20	-0,05	0,04
	5	0,95	0,05	80	5	2	0,25	-0,05	0,05
	6	0,75	0,10	100	5	2	0,05	-0,10	0,06
	7	0,80	0,10	60	8	2	0,10	-0,10	0,07
	8	0,85	0,10	80	8	2	0,15	-0,10	0,08
	9	0,90	0,10	100	8	2	0,20	-0,10	0,09
	10	0,95	0,10	60	2	5	0,25	-0,10	0,10
	11	0,75	0,15	80	2	5	0,05	-0,15	0,11
	12	0,80	0,15	100	2	5	0,10	-0,15	0,12
	13	0,85	0,15	60	5	5	0,15	-0,15	0,13
	14	0,90	0,15	80	5	5	0,20	-0,15	0,14
	15	0,95	0,15	100	5	5	0,25	-0,15	0,15
	16	0,75	0,20	60	8	5	0,05	-0,20	0,16
	17	0,80	0,20	80	8	5	0,10	-0,20	0,17
	18	0,85	0,20	100	8	5	0,15	-0,20	0,18
	19	0,90	0,20	60	2	8	0,20	-0,20	0,19
	20	0,95	0,20	80	2	8	0,25	-0,20	0,20
	21	0,75	0,25	100	2	8	0,05	-0,25	0,21
	22	0,80	0,25	60	5	8	0,10	-0,25	0,22
	23	0,85	0,25	80	5	8	0,15	-0,25	0,23
	24	0,90	0,25	100	5	8	0,20	-0,25	0,24
	25	0,95	0,25	60	8	8	0,25	-0,25	0,25
	26			80	8	8			
	27			100	8	8			

Para discussão sobre a influência de valores utilizados em cada parâmetro específico no número de segmentos obtidos, a Tabela 9 apresenta coeficientes de correlação calculados entre o número médio de segmentos obtido por classe de movimento e parâmetros utilizados.

Tabela 9 – Coeficientes de correlação entre parâmetros utilizados e média de número de segmentos obtido por classe de movimento.

Método#	1		2			3		4
Base de Dados	q	$T_{lim}$	A	B	C	B	C	T
NinaPro	0,11	-0,97	-0,38	-0,27	0,77	-0,42	-0,66	-0,35
IEE	0,07	-0,98	-0,26	-0,18	0,89	-0,32	0,64	0,80

Com os métodos de segmentação MTD1 e MTD2, percebe-se que a moda de número de segmentos por classe de movimento manteve-se de forma consistente em 6 para uma grande variedade de combinações de parâmetros utilizados em ambas as bases de dados.

No MTD1, pode-se perceber forte correlação entre a média de número de segmentos obtidos por classe e o valor utilizado para  $T_{lim}$  - combinações com maior  $T_{lim}$  apresentaram menor média de número de segmentos obtidos por classe de movimento. Isto pode ser justificado pelo fato de  $T_{lim}$  ser também um parâmetro para interrupção do método iterativo do MTD1 (quanto maior  $T_{lim}$ , menor o número máximo de iterações do método e por consequência, menor o número de segmentos identificados) e indica que para a maioria das execuções do MTD1 o limite mínimo de *threshold*  $T_{lim}$  é atingido antes de obter-se a razão mínima desejada  $r_{target}$  entre número de segmentos e comprimento total de sinal.

Para o MTD2, evidenciou-se forte correlação entre o número médio de segmentos obtidos por classe de movimento e o parâmetro  $C$ , indicando que a escolha de fração do valor máximo do sinal como valor de *threshold* como tem maior influência no número de segmentos obtidos que a escolha de um múltiplo da média do sinal.

Para o MTD3 e o MTD4, apenas poucas combinações de parâmetros testadas conseguiram apresentar moda 6 para número de segmentos por classe de movimento na base de dados NinaPro (5 combinações com MTD3, 8 combinações com MTD4) e nenhuma combinação de parâmetros atingiu moda 6 na base de dados IEE.

Para o MTD3, em ambas as bases de dados, evidencia-se correlações moderadas entre número médio de segmentos obtidos e os parâmetros utilizados, sendo que para o parâmetro  $C$  o coeficiente de correlação apresentou sinais opostos para diferentes bases de dados. Resultados para correlação neste método são, portanto, inconclusivos.

Para o MTD4 existe forte correlação positiva entre o valor de *threshold* utilizado e o número de segmentos obtido para a base de dados IEE; e correlação moderada-fraca de sinal oposto para a base de dados NinaPro.

## 4.2 Segmentação e Classificação por Classes de Movimento

Para avaliação de desempenho de classificação das diferentes classes de movimento em cada método de segmentação, foram selecionadas as combinações de parâmetros apresentadas na Tabela 10, que, dentre as combinações testadas, apresentaram o número médio de segmentos por classe de movimento mais próximo de 6.

Tabela 10 – Combinações de parâmetros selecionados para cada base de dados.

Método	Base de dados	Índice	Parâmetros			Média de Segmentos por Classe	Moda de Segmentos por Classe
MTD1	NinaPro	15	$q = 0.95$	$T_{lim} = 0.15$		6,05	6
	IEE	11	$q = 0.75$	$T_{lim} = 0.15$		6,23	6
MTD2	NinaPro	7	$A = 60$	$B = 8$	$C = 2$	6,01	6
	IEE	18	$A = 100$	$B = 8$	$C = 5$	6,28	6
MTD3	NinaPro	7	$B = 0.10$	$C = -0.10$		4,29	6
	IEE	7	$B = 0.10$	$C = -0.10$		3,15	3
MTD4	NinaPro	6		$T = 0.06$		4,46	6
	IEE	21		$T = 0.21$		2,78	2

A Figura 20 e a Tabela 11 apresentam resultados de valores  $\bar{F}_\zeta$  (Equação (2.11)), indexados nas classes  $\zeta$  de acordo com a Tabela 6, utilizando os parâmetros selecionados na Tabela 10.

Para a base de dados NinaPro, os piores desempenhos de classificação entre métodos foram verificados para a classe de movimento 1 (polegar esticado) e 7 (extensão do indicador), movimentos que realizam a extensão de apenas um dedo da mão e flexão dos demais. Os melhores desempenhos para esta base foram verificados nas classes 6 (flexão de todos os dedos ao punho), 14 (extensão de punho), 16 (desvio ulnar do punho) e 17 (extensão de punho com mão cerrada).

Para a base de dados IEE, foi identificada grande dificuldade de classificação (valor  $F$  médio inferior a 0,7) para as classes 8 (adução de dedos) e 10 (rotação de punho em torno do dedo médio). Os melhores desempenhos de classificação desta base foram obtidos para as classes 5 (abdução de dedos) e 16 (desvio ulnar do punho).

Figura 16 – Média e moda do número de segmentos obtidos por classe de movimento e valor  $F$  médio de RNAs com diferentes combinações de parâmetros. Método de segmentação: MTD1.

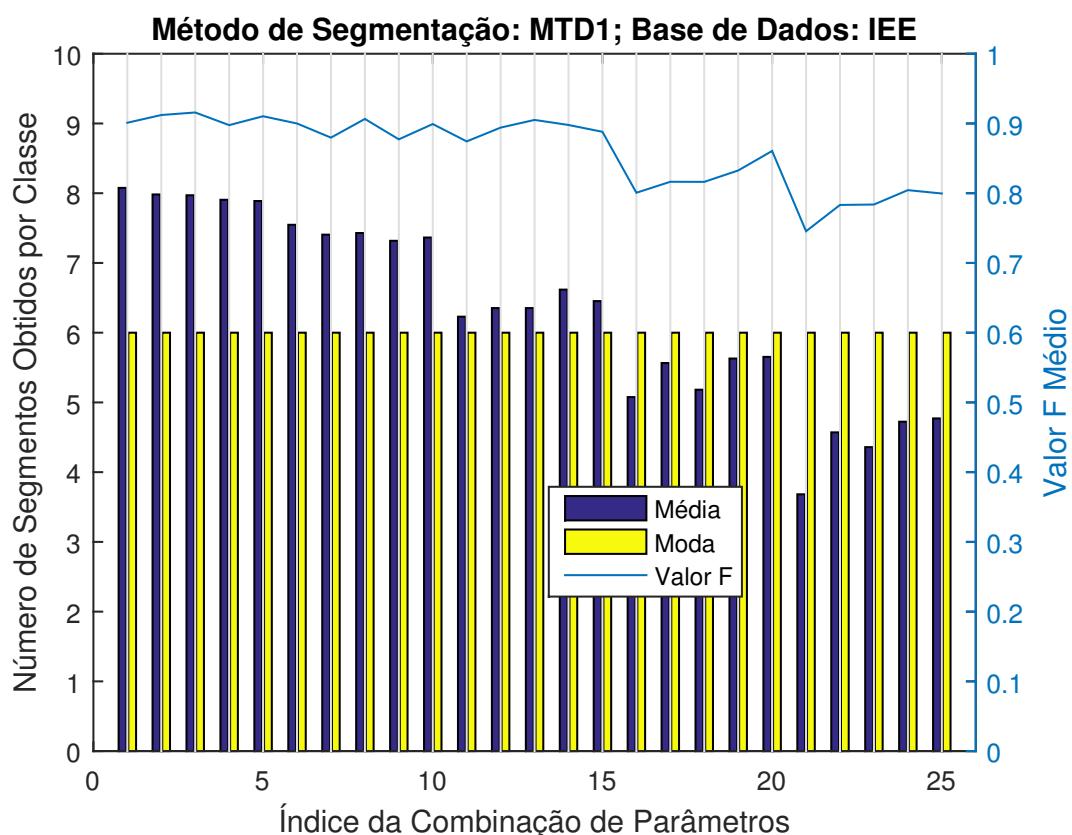
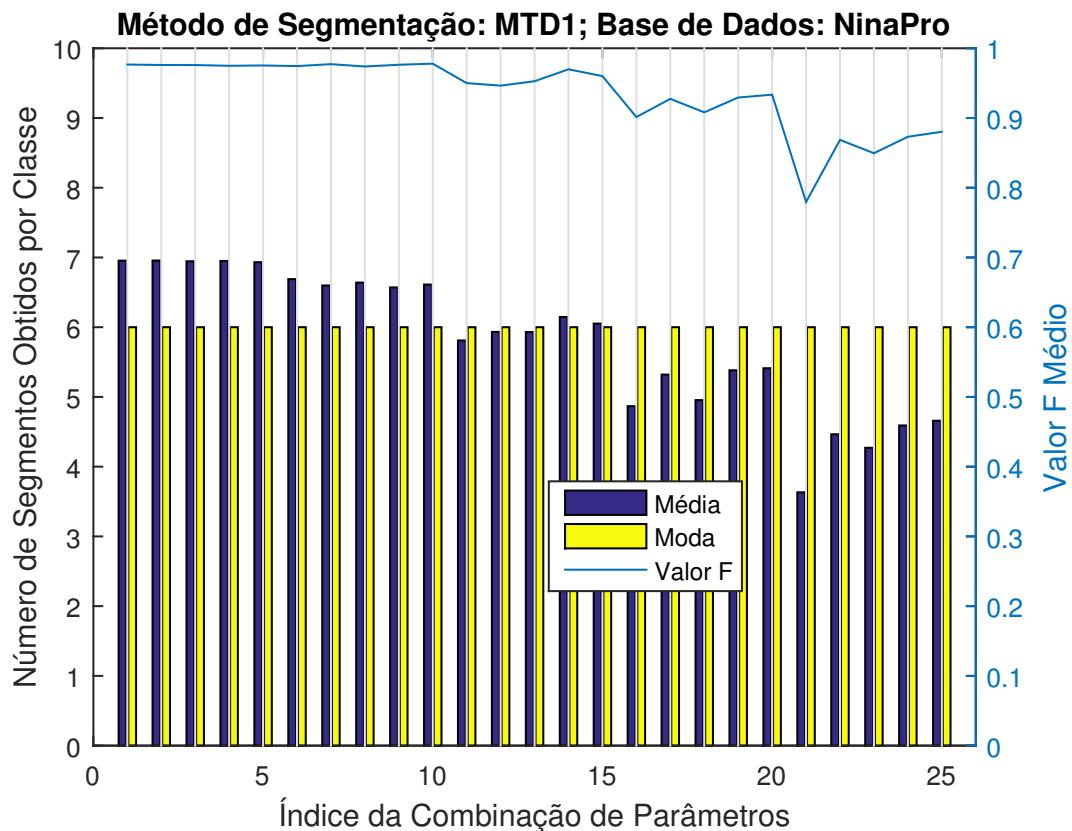


Figura 17 – Média e moda do número de segmentos obtidos por classe de movimento e valor  $F$  médio de RNAs com diferentes combinações de parâmetros. Método de segmentação: MTD2.

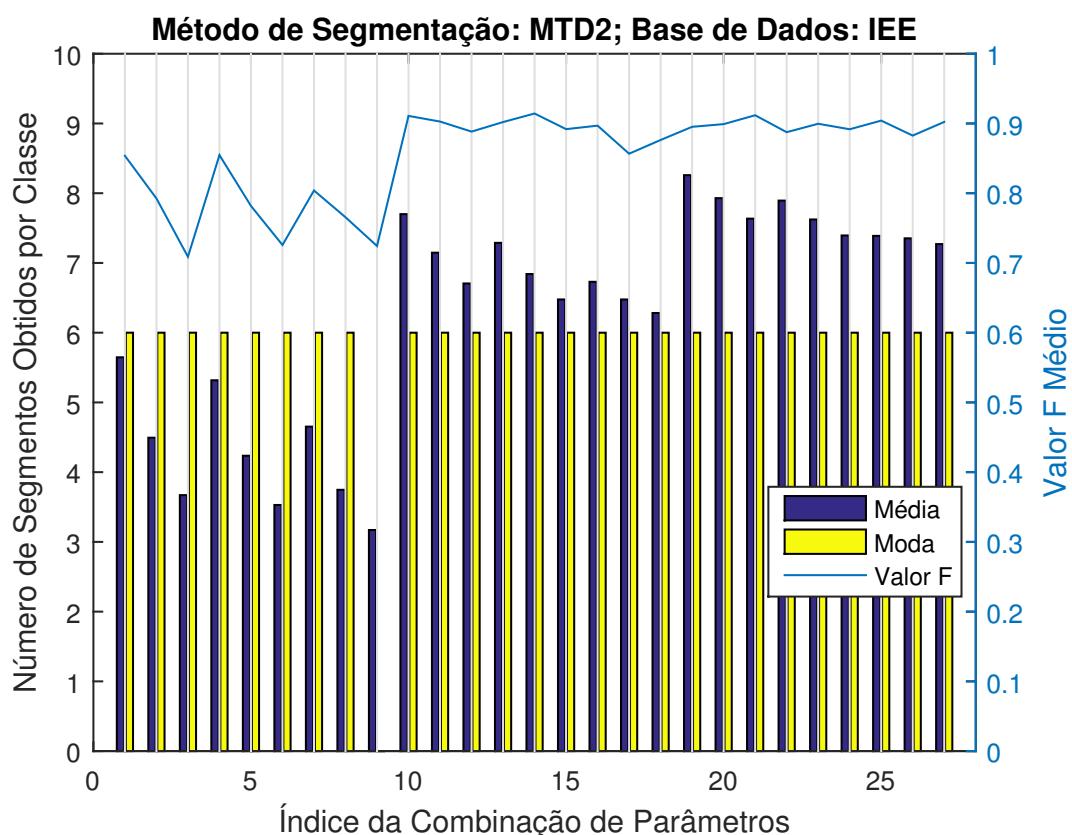
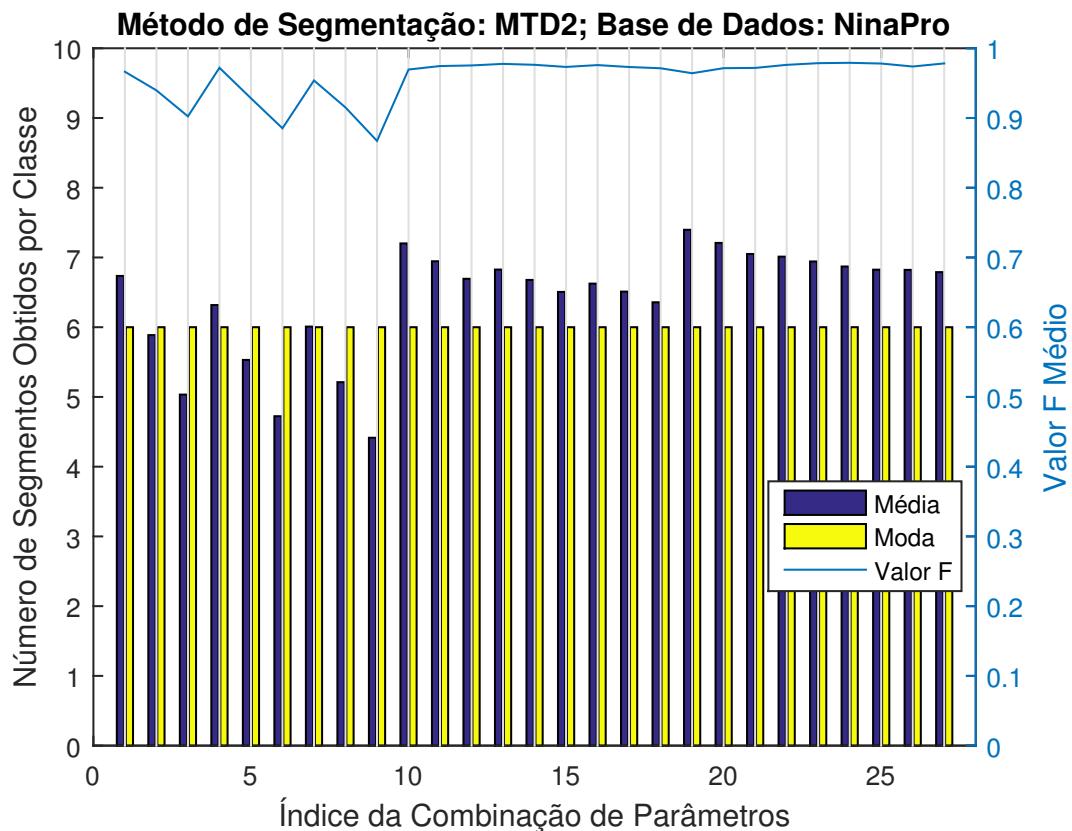


Figura 18 – Média e moda do número de segmentos obtidos por classe de movimento e valor  $F$  médio de RNAs com diferentes combinações de parâmetros. Método de segmentação: MTD3.

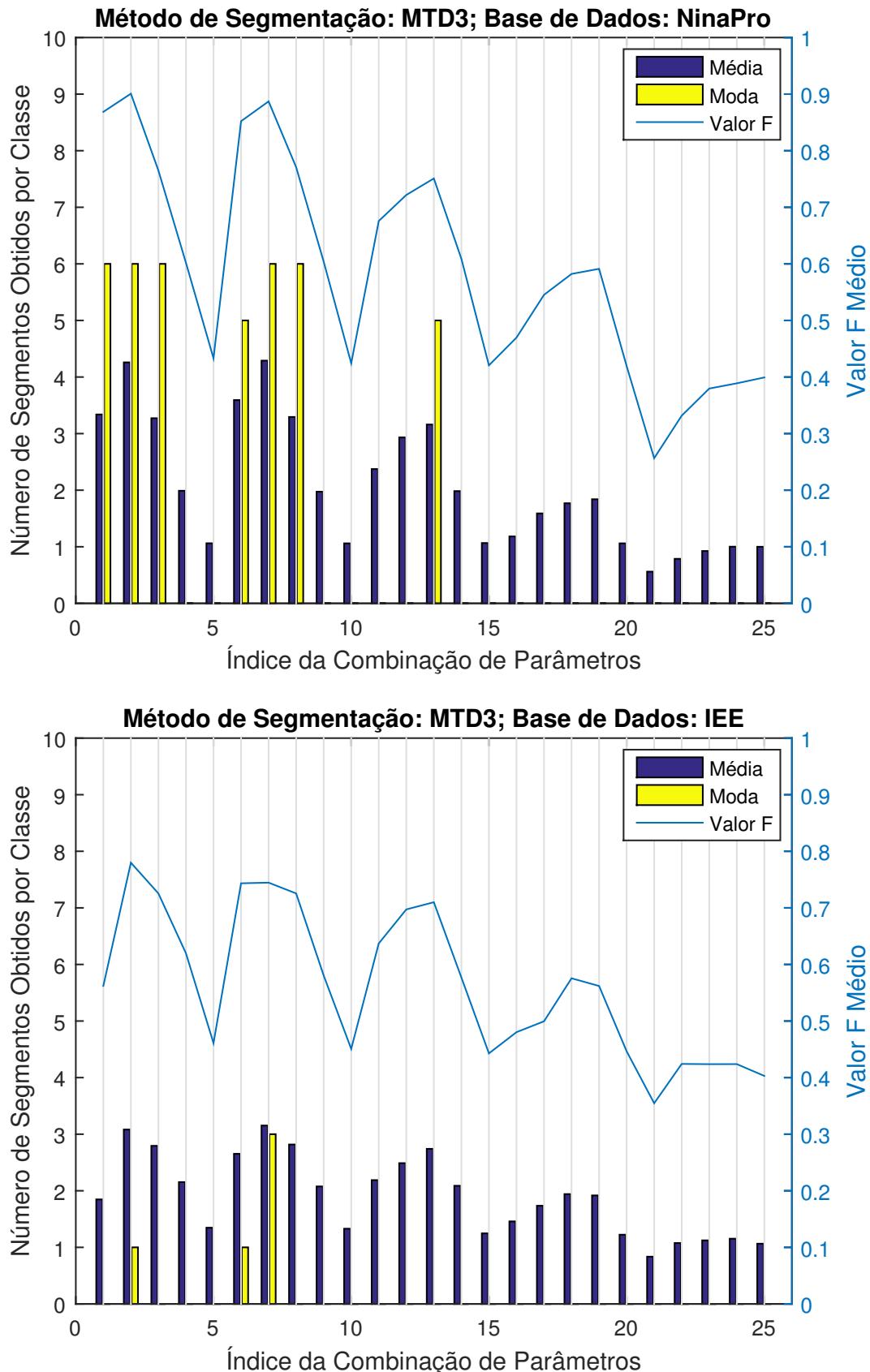


Figura 19 – Média e moda do número de segmentos obtidos por classe de movimento e valor  $F$  médio de RNAs com diferentes combinações de parâmetros. Método de segmentação: MTD4.

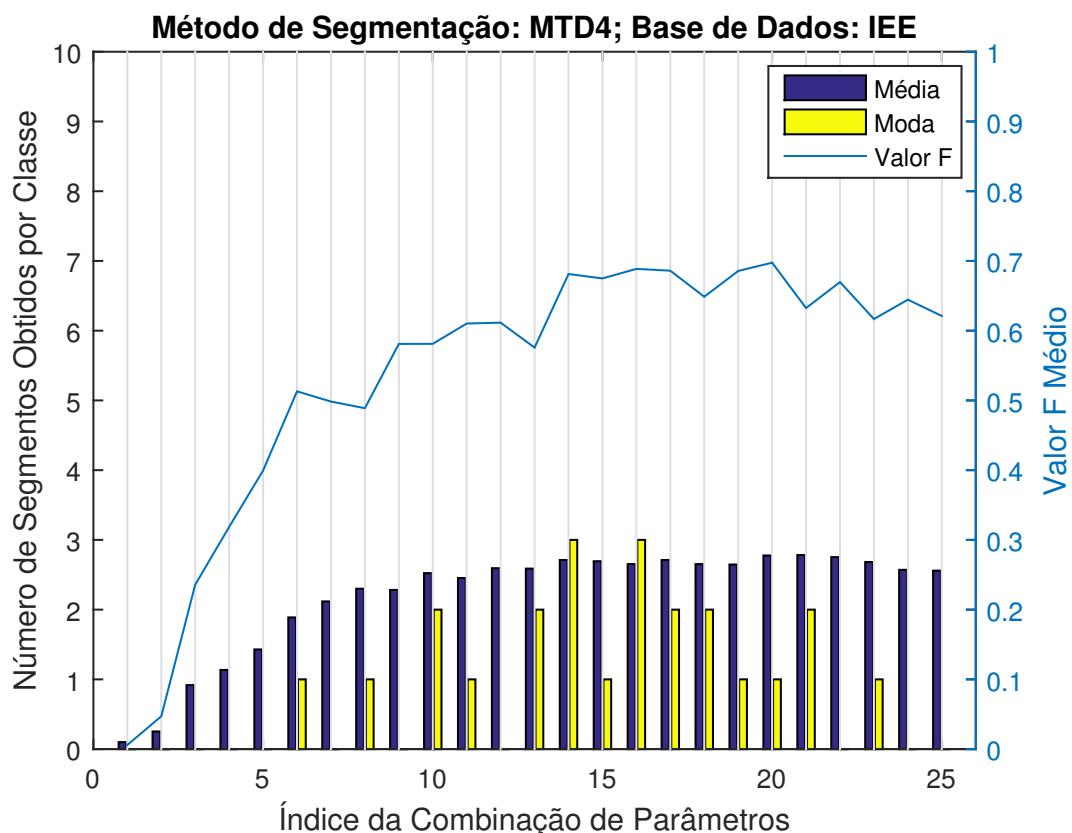
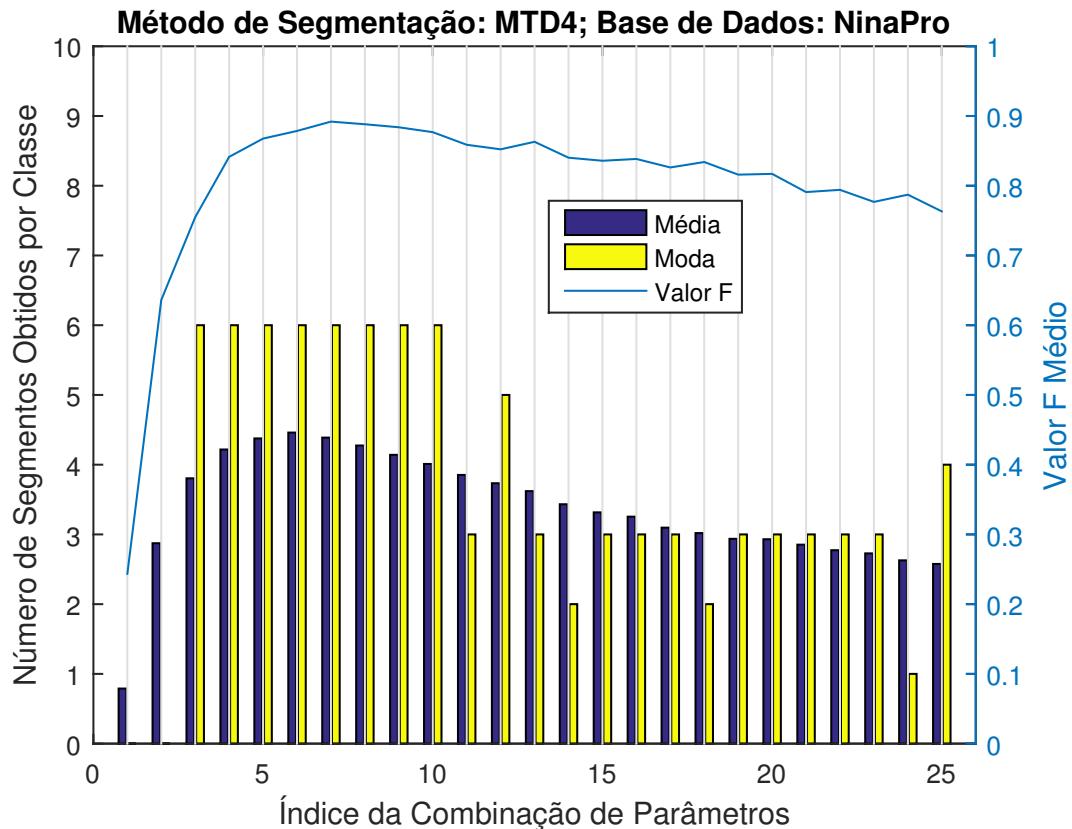


Figura 20 – Valor  $F$  médio por classe de movimento para cada base de dados, utilizando parâmetros selecionados em cada método de segmentação.

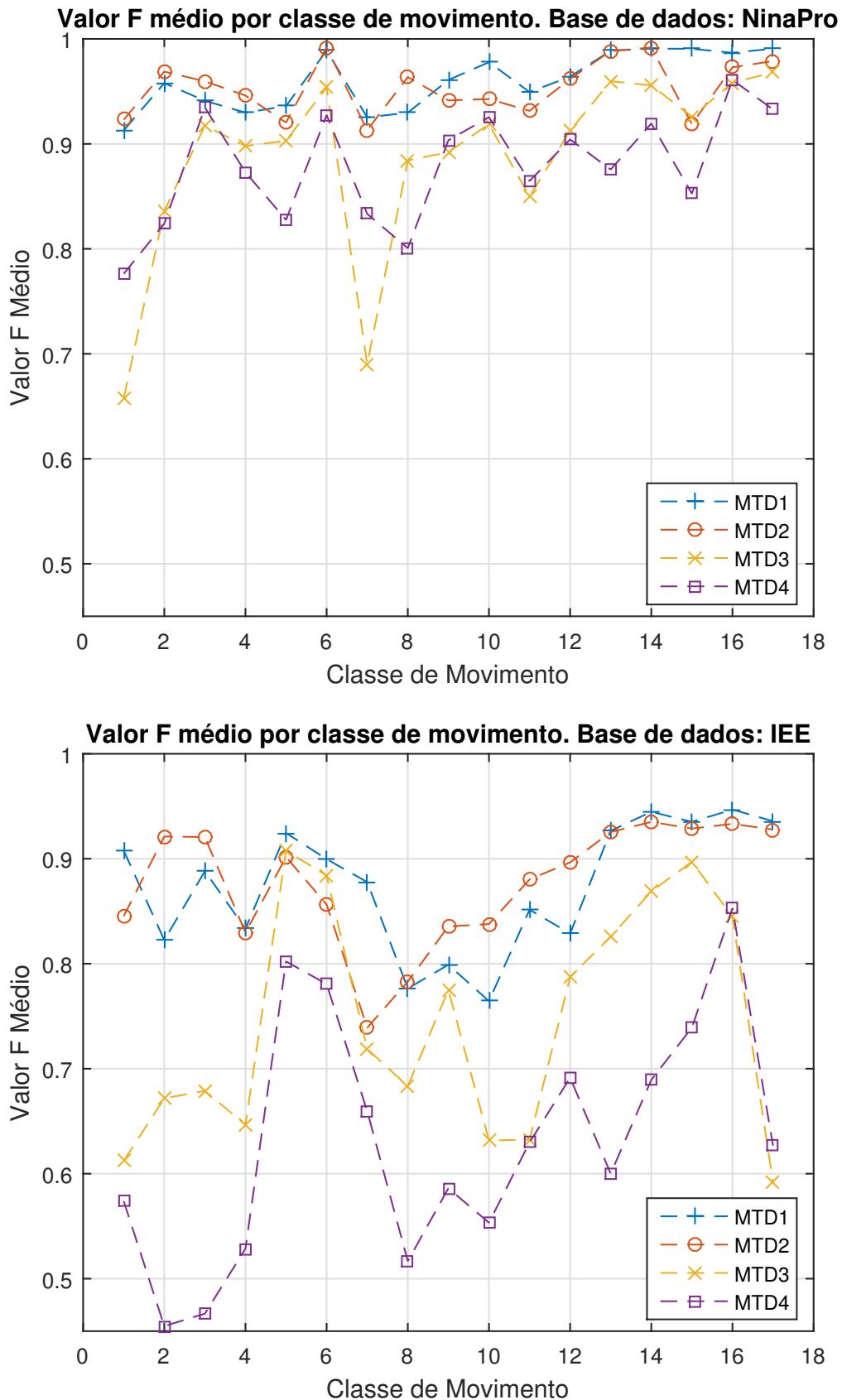


Tabela 11 – Valor  $F$  médio por classe de movimento utilizando parâmetros selecionados em cada método

		Classe de Movimento																			
		Base de Dados		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	Média
MTD1	NinaPro	0,91	0,95	0,94	0,92	0,93	0,98	0,92	0,92	0,97	0,94	0,96	0,98	0,99	0,99	0,98	0,99	0,99	0,99	0,96	
	IEE	0,90	0,82	0,88	0,83	0,92	0,89	0,87	0,77	0,79	0,76	0,85	0,82	0,92	0,94	0,93	0,94	0,93	0,93	0,87	
MTD2	NinaPro	0,92	0,96	0,95	0,94	0,92	0,99	0,91	0,96	0,94	0,94	0,93	0,96	0,98	0,99	0,99	0,91	0,97	0,97	0,97	0,95
	IEE	0,84	0,92	0,92	0,82	0,90	0,85	0,73	0,78	0,83	0,83	0,88	0,89	0,92	0,93	0,92	0,93	0,92	0,92	0,87	0,88
MTD3	NinaPro	0,65	0,83	0,91	0,89	0,90	0,95	0,69	0,88	0,89	0,91	0,84	0,91	0,95	0,95	0,92	0,95	0,95	0,96	0,96	0,88
	IEE	0,61	0,67	0,67	0,64	0,90	0,88	0,71	0,68	0,77	0,63	0,63	0,78	0,82	0,86	0,86	0,89	0,84	0,59	0,74	0,88
MTD4	NinaPro	0,77	0,82	0,93	0,87	0,82	0,92	0,83	0,80	0,90	0,92	0,86	0,90	0,87	0,91	0,87	0,91	0,85	0,96	0,93	0,87
	IEE	0,57	0,45	0,46	0,52	0,80	0,78	0,65	0,51	0,58	0,55	0,62	0,69	0,60	0,68	0,73	0,85	0,62	0,63	0,63	0,87
Média	NinaPro	0,81	0,89	0,93	0,91	0,89	0,96	0,84	0,89	0,92	0,94	0,89	0,93	0,95	0,96	0,92	0,96	0,96	0,96	0,91	0,91
	IEE	0,73	0,71	0,73	0,70	0,88	0,85	0,74	0,68	0,74	0,69	0,74	0,69	0,80	0,81	0,85	0,87	0,77	0,78	0,77	0,78

Em **vermelho**, valores  $F$  médio inferiores a 0,80; em **verde**, valores  $F$  médio superiores a 0,95.

## 5 Conclusões

Os métodos de segmentação baseados na detecção de centros de segmentos com segmentos de comprimento constante (MTD1 e MTD2) foram responsáveis por melhores resultados de classificação (valores  $F$  médio cerca de 17% maiores) que métodos de segmentação que detectam BEPs e EEPs para segmentos de comprimento variável (MTD3 e MTD4) em ambas as bases de dados.

Os valores  $F$  médio obtidos com MTD2 foram similares a quando foi utilizado MTD1, mostrando que o cálculo de *threshold* de forma iterativa não apresenta melhorias significativas no resultado final que o uso de um *threshold* calculado uma única vez. Ao considerarmos o custo computacional de métodos, o MTD2 apresenta clara vantagem sobre o MTD1 ao obter resultados equivalentes com menor número de cálculos realizados.

Com o uso de RNAs para classificação na base de dados NinaPro, evidencia-se que movimentos com flexão de todos os dedos (“fechamento” de punho) estão entre as classes com melhor desempenho de classificação, enquanto os piores resultados são para flexão de quatro dedos e extensão de único dedo. Isto mostra que, para o posicionamento de eletrodos utilizado na aquisição desta base de dados, movimentos de flexão de múltiplos dedos influenciam fortemente os sinais de EMG adquiridos e a extensão de um único dedo passa a ser de difícil classificação. Obteve-se bons resultados para a classificação de desvio ulnar do punho para as duas bases de dados.

Para métodos utilizando janela deslizante e comprimentos de segmentos variável (MTD3 e MTD4) não foi possível atingir moda 6 de número de segmentos por classe na base IEE com nenhuma combinação de parâmetros testada, mas isto foi possível para a base NinaPro. Tais diferenças entre bases de dados possivelmente são justificados por um maior nível de ruído na aquisição do IEE ou os conjuntos de parâmetros testados para MTD3 e MTD4 não incluem valores adequados para esta base de dados.

Dentre os métodos de segmentação automática propostos e testados neste trabalho, o método de segmentação não iterativo utilizando *threshold* para detecção de centros de segmentos de comprimento constante (MTD2) apresentou os resultados mais satisfatórios para obtenção de segmentos e classificação de movimentos.

## 6 Propostas de Trabalhos Futuros

Sugere-se proposição de métodos de segmentação que apresentem resultados melhores para as aquisições realizadas no IEE. Caso o nível de ruído de aquisição justifique os resultados comparativamente piores com relação à base NinaPro, pode-se explorar métodos de segmentação baseados em “envelopes” de sinal (a exemplo de D’Alessio e Conforto (2001) e Sedlak et al. (2013)) ou outros critérios de detecção de BEPs e EEPs (e.g. valores limite para RMS do sinal janelado).

Pode-se explorar novas características de segmentos utilizadas como preditores de RNA, em diferentes possibilidades de combinações. Phinyomark, Phukpattaranont e Limsakul (2012) listam um vasto número de possíveis características a serem exploradas.

Caso um trabalho futuro tenha como foco aplicações de controle de próteses mioelétricas, os métodos necessitariam ser reescritos para versões completamente causais. Algumas estratégias que contornam a necessidade de detectar-se todo o segmento de movimento para depois classificá-lo envolvem a constante extração de características de janelas do sinal e classificação dos mesmos, considerando a não-realização de movimento como também possível classe, a exemplo de Englehart e Hudgins (2003).

## Referências Bibliográficas

- ABRAHAM, A. Artificial neural networks. In: *Handbook of Measuring System Design.* [S.l.]: John Wiley & Sons, Ltd, 2005. Citado na página 24.
- ALMEIDA, M.; BARRETO, J. *Filtragem digital de sinais biomédicos.* [S.l.]: Universidade Federal de Santa Catarina, Centro Tecnológico, 1997. Citado na página 16.
- ATZORI, M. et al. Electromyography data for non-invasive naturally-controlled robotic hand prostheses. *Scientific Data*, v. 1, Dec 2014. Citado na página 33.
- BARBOSA, A. H.; FREITAS, M. S. d. R.; NEVES, F. d. A. d. Confiabilidade estrutural utilizando o método de monte carlo e redes neurais. *Rem: Revista Escola de Minas*, v. 58, p. 247 – 255, 9 2005. Citado na página 28.
- BASMAJIAN, J.; LUCA, C. D. *Muscles Alive.* 5. ed. Baltimore: Williams and Wilkins, 1985. Citado na página 16.
- BU, N.; FUKUDA, O. Emg-based motion discrimination using a novel recurrent neural network. *J. Intell. Inf. Syst.*, 2003. Citado na página 23.
- BUCHTHAL, F.; SCHMALBRUCH, H. Motor unit of mammalian muscle. *Physiological Reviews*, v. 60, n. 1, p. 90–142, 1980. Citado na página 16.
- CENE, V. H.; BALBINOT, A. Upper-limb movement classification based on sEMG signal validation with continuous channel selection. In: LA-CCI. *Annals of 2nd Latin-American Congress on Computational Intelligence.* Curitiba, PR, 2015. ISBN: 978-85-69972-00-6. Citado na página 36.
- CHAN, F. H. et al. Fuzzy EMG classification for prosthesis control. *IEEE Trans Rehabil Eng*, v. 8, n. 3, p. 305–311, Sep 2000. Citado na página 14.
- CHAUVET, E. et al. A method of EMG decomposition based on fuzzy logic. In: *Engineering in Medicine and Biology Society, 2001. Proceedings of the 23rd Annual International Conference of the IEEE.* [S.l.: s.n.], 2001. v. 2, p. 1948–1950. Citado 2 vezes nas páginas 14 e 18.
- CHU, J.-U. et al. A supervised feature-projection-based real-time emg pattern recognition for multifunction myoelectric hand control. *Mechatronics, IEEE/ASME Transactions on*, v. 12, n. 3, p. 282–290, June 2007. Citado na página 14.
- D'ALESSIO, T.; CONFORTO, S. Extraction of the envelope from surface EMG signals. *Engineering in Medicine and Biology Magazine, IEEE*, Nov, n. 6, p. 55–61, 7 2001. Citado na página 49.
- DREW, P. J.; MONSON, J. R. Neural networks. *Surgery*, v. 127, n. 1, p. 201–213, 3 -11 2000. Citado na página 24.
- ENGLEHART, K.; HUDGINS, B. A robust, real-time control scheme for multifunction myoelectric control. *Biomedical Engineering, IEEE Transactions on*, v. 50, n. 7, p. 848–854, July 2003. Citado 2 vezes nas páginas 14 e 49.

- ESTER, M. et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In: . [S.l.]: AAAI Press, 1996. p. 226–231. Citado 2 vezes nas páginas 34 e 35.
- FAVIEIRO, G.; BALBINOT, A. Adaptive neuro-fuzzy logic analysis based on myoelectric signals for multifunction prosthesis control. In: *Engineering in Medicine and Biology Society, EMBC, 2011 Annual International Conference of the IEEE*. [S.l.: s.n.], 1993. p. 7888–7891. Citado na página 14.
- FAVIEIRO, G. W. *Controle de uma Prótese Experimental do Segmento Mão-Braço por Sinais Mioelétricos e Redes Neurais Artificiais*. 2009. Trabalho de Diplomação, Curso de Engenharia de Computação, UFRGS. Citado 2 vezes nas páginas 17 e 36.
- GERDLE, B. et al. Acquisition, processing and analysis of the surface electromyogram. In: *Modern Techniques in Neuroscience Research*. [S.l.]: Springer Berlin Heidelberg. Citado na página 17.
- GIJSBERTS, A. et al. Measuring movement classification performance with the movement error rate. *IEEE Transactions on neural systems and rehabilitation engineering*, 2014. Citado 3 vezes nas páginas 29, 30 e 31.
- GUT, R.; MOSCHYTZ, G. S. High-precision EMG signal decomposition using communication techniques. *Signal Processing, IEEE Transactions on*, v. 48, n. 9, p. 2487–2494, set. 2000. Citado 3 vezes nas páginas 15, 20 e 21.
- HARGROVE, L. J. et al. Robotic leg control with emg decoding in an amputee with nerve transfers. *New England Journal of Medicine*, v. 369, n. 13, p. 1237–1242, 2013. Citado na página 14.
- HIRAIWA, A.; SHIMOHARA, K.; TOKUNAGA, Y. EMG pattern analysis and classification by neural network. In: *Systems, Man and Cybernetics, 1989. Conference Proceedings., IEEE International Conference on*. [S.l.: s.n.], 1989. v. 3, p. 1113–1115. Citado na página 23.
- HORNIK, K. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, v. 4, n. 4, p. 251 – 257, 1991. Citado na página 23.
- HUDGINS, B.; PARKER, P.; SCOTT, R. A new strategy for multifunction myoelectric control. *Biomedical Engineering, IEEE Transactions on*, v. 1, n. 11, p. 82–94, jan. 1993. Citado 2 vezes nas páginas 14 e 23.
- KATSIS, C. et al. A novel method for automated EMG decomposition and MUAP classification. *Artificial Intelligence in Medicine*, v. 37, n. 1, p. 55–64, 2006. Citado 2 vezes nas páginas 14 e 20.
- KIM, K. jae; HA, I. Genetic algorithms approach to feature discretization in artificial neural networks for the prediction of stock price index. *Expert Systems with Applications*, v. 19, n. 2, p. 125–132, 2000. Citado na página 24.
- KLINE, J. C.; LUCA, C. J. D. Error reduction in EMG signal decomposition. *Journal of Neurophysiology*, 2014. Citado na página 17.

- KUBAT, M.; HOLTE, R.; MATWIN, S. Machine learning for the detection of oil spills in satellite radar images. *Machine Learning*, v. 30, n. 2-3, p. 195–215, 1998. Citado na página 25.
- LOPES, I. F. *Caracterização dos Sinais Mioelétricos dos Movimentos do Segmento Mão-Braço Através de Regressão Logística*. 2014. Trabalho de Diplomação, Curso de Engenharia de Computação, UFRGS. Citado na página 17.
- LUCA, C. J. D. et al. Decomposition of surface EMG signals. *J. Neurophysiol.*, v. 96, n. 3, p. 1646–1657, set. 2006. Citado na página 16.
- MØLLER, M. F. A scaled conjugate gradient algorithm for fast supervised learning. *Neural Networks*, v. 6, n. 4, p. 525–533, 1993. Citado na página 24.
- PATTICHIS, C.; SCHIZAS, C.; MIDDLETON, L. Neural network models in emg diagnosis. *Biomedical Engineering, IEEE Transactions on*, v. 42, n. 5, p. 486–496, maio 1995. Citado 4 vezes nas páginas 14, 15, 18 e 22.
- PHINYOMARK, A.; PHUKPATTARANONT, P.; LIMSAKUL, C. Feature reduction and selection for EMG signal classification. *Expert Systems with Applications*, v. 39, n. 8, p. 7420 – 7431, 2012. Citado na página 49.
- SCHONS, L. *Caracterização dos Sinais Mioelétricos dos Movimentos do Segmento Mão-Braço Através de Redes Neurais Artificiais e Algoritmos Genéticos*. 2014. Trabalho de Diplomação, Curso de Engenharia de Computação, UFRGS. Citado 2 vezes nas páginas 17 e 36.
- SEDLAK, J. et al. Segmentation of surface emg signals. In: *Applied Electronics (AE), 2013 International Conference on*. [S.l.: s.n.], 2013. p. 1–4. Citado na página 49.
- SUBASI, A.; YILMAZ, M.; OZCALIK, H. R. Classification of EMG signals using wavelet neural network. *Journal of Neuroscience Methods*, n. 1-2, p. 360 – 367, 2006. Citado na página 23.
- TANIKIĆ, V. D. D. Metallurgy - Advances in Materials and Processes. 2012. Citado 2 vezes nas páginas 23 e 28.
- TRAN, T. N.; DRAB, K.; DASZYKOWSKI, M. Revised DBSCAN algorithm to cluster data with dense adjacent clusters. *Chemometrics and Intelligent Laboratory Systems*, v. 120, p. 92–96, 2013. Citado 5 vezes nas páginas 13, 35, 75, 76 e 77.
- YEGNANARAYANA, B. *Artificial Neural Networks*. [S.l.]: PHI Learning, 2009. ISBN 9788120312531. Citado 2 vezes nas páginas 23 e 24.

# Apêndices

# APÊNDICE A – Função em MATLAB para MTD1

```

function [x_seg, finalCenterLocs] = seg_mtd1(x, l, q, r_target, T_lim)
% MTD1 - metodo iterativo utilizando thresholding para deteccao de
% centros de segmentos de comprimento constante
%
% Argumentos: (para mais detalhes, refira a descricao do MTD1)
%   x - matriz cujas colunas sao canais do sinal a ser segmentado
%   l - comprimento desejado para os segmentos
%   q - razao de atualizacao entre iteracoes para valor de threshold
%   r_target - razao minima esperada entre numero de segmentos e comprimento
%             total de sinal
%   T_lim - fracao do maximo do sinal para limite inferior de threshold
%
% Retorno:
%   x_seg - cell array com canais segmentados
%   finalCenterLocs - posicoes centrais dos segmentos

%% Preprocessamento

[L, numberOfChannels] = size(x); % comprimento do sinal e numero de canais
x_ret = abs(x); % retificacao
x_norm = zeros(L, numberOfChannels); % normalizacao
for currentChannel = 1:numberOfChannels
    x_norm(:,currentChannel) = ...
        x_ret(:,currentChannel)./max(x_ret(:,currentChannel));
end

%% Metodo

centerLocsCell = cell(1,numberOfChannels);
for currentChannel = 1:numberOfChannels
    T_k = 1; % canais normalizados, seus valores maximos sempre sao 1
    targetReached = false;
    while ~targetReached % processo iterativo
        T_k = q*T_k; % calcula threshold desta iteracao
        if T_k < T_lim % verifica se o limite de valor de threshold foi atingido
            break
        end
        % Identifica candidatos a centros de segmentos
        [~, centerLocsCell{1,currentChannel}] = ...

```

```

        findpeaks(x_norm(:,currentChannel), ...
        'MinPeakHeight', T_k, 'MinPeakDistance',1);
    % Determina o encerramento do processo iterativo
    targetReached = (length(centerLocsCell{1,currentChannel})/L >= r_target);
end

%% Clustering

centerLocsArray = sort(cell2mat(centerLocsCell'));
[~, labscore] = dbscan(centerLocsArray,2000,3);
numberOfSegments = max(labscore);
finalCenterLocs = zeros(numberOfSegments,1); % medias internas aos clusters
for currentCluster = 1:numberOfSegments
    finalCenterLocs(currentCluster) = ...
        round(mean(centerLocsArray(labscore == currentCluster)));
end

%% Segmentacao

x_seg = cell(numberOfSegments, numberOfChannels);
for currentChannel = 1:numberOfChannels
    for currentSegment = 1:numberOfSegments
        switch mod(l,2)
            case 0 % se l for par
                if(finalCenterLocs(currentSegment)-l/2)<1
                    % segmento muito a esquerda
                    x_seg{currentSegment,currentChannel} = ...
                        x(1:finalCenterLocs(currentSegment)+(l/2)-1, ...
                        currentChannel);
                else if(finalCenterLocs(currentSegment)+(l/2)-1)>L
                    % segmento muito a direita
                    x_seg{currentSegment,currentChannel} = ...
                        x(finalCenterLocs(currentSegment)-l/2:L, ...
                        currentChannel);
                else
                    x_seg{currentSegment,currentChannel} = ...
                        x(finalCenterLocs(currentSegment)-l/2: ...
                        finalCenterLocs(currentSegment)+(l/2)-1, ...
                        currentChannel);
                end
            end
            case 1 % se l for impar
                if(finalCenterLocs(currentSegment) - (l-1)/2)<1
                    % segmento muito a esquerda
                    x_seg{currentSegment,currentChannel} = ...
                        x(1:finalCenterLocs(currentSegment) + (l-1)/2, ...

```

```
        currentChannel);
else if(finalCenterLocs(currentSegment) + (l-1)/2>L
    % segmento muito a direita
x_seg{currentSegment,currentChannel} = ...
    x(finalCenterLocs(currentSegment) - (l-1)/2:L, ...
        currentChannel);
else
    x_seg{currentSegment,currentChannel} = ...
        x(finalCenterLocs(currentSegment) - (l-1)/2: ...
            finalCenterLocs(currentSegment) + (l-1)/2, ...
                currentChannel);
end
end
end
end
end
```

# APÊNDICE B – Função em MATLAB para MTD2

```

function [x_seg, finalCenterLocs] = seg_mtd2(x, l, A, B, C)
% MTD2 - metodo nao iterativo utilizando thresholding para deteccao de
% centros de segmentos de comprimento constante
%
% Argumentos: (para mais detalhes, refira a descricao do MTD2)
%   x - matriz cujas colunas sao canais do sinal a ser segmentado
%   l - comprimento desejado para os segmentos
%   A - coeficiente utilizado para decisao de metodo de calculo de threshold
%   B - multiplo da media aritmetica do sinal x para obtencao de threshold
%   C - fracao do valor maximo do sinal x para calculo de threshold
%
% Retorno:
%   x_seg - cell array com os canais segmentados
%   finalCenterLocs - posicoes centrais dos segmentos

%% Preprocessamento

[L, numberOfChannels] = size(x); % comprimento do sinal e numero de canais
x_ret = abs(x); % retificacao
x_norm = zeros(L, numberOfChannels); % normalizacao
for currentChannel = 1:numberOfChannels
    x_norm(:,currentChannel) = ...
        x_ret(:,currentChannel)./max(x_ret(:,currentChannel));
end

%% Metodo

centerLocsCell = cell(1,numberOfChannels);
for currentChannel = 1:numberOfChannels
    % Calculo do threshold
    maxValue = 1; % sinais normalizados
    meanValue = mean(x_norm(:,currentChannel));
    if maxValue > (A*meanValue)
        T = B*meanValue;
    else
        T = maxValue/C;
    end
    % Identifica centros de segmentos
    [~, centerLocsCell{1,currentChannel}] = ...

```

```

    findpeaks(double(x_norm(:,currentChannel)), ...
    'MinPeakHeight', T, 'MinPeakDistance',1);
end

%% Clustering

centerLocsArray = sort(cell2mat(centerLocsCell'));
[~, labscore] = dbscan(centerLocsArray,2000,3);
numberOfSegments = max(labscore);
finalCenterLocs = zeros(numberOfSegments,1); % medias internas aos clusters
for currentCluster = 1:numberOfSegments
    finalCenterLocs(currentCluster) = ...
        round(mean(centerLocsArray(labscore == currentCluster)));
end

%% Segmentacao

x_seg = cell(numberOfSegments, numberOfChannels);
for currentChannel = 1:numberOfChannels
    for currentSegment = 1:numberOfSegments
        switch mod(l,2)
            case 0 % se l for par
                if(finalCenterLocs(currentSegment)-l/2)<1
                    % segmento muito a esquerda
                    x_seg{currentSegment,currentChannel} = ...
                        x(1:finalCenterLocs(currentSegment)+(l/2)-1, ...
                        currentChannel);
                else if(finalCenterLocs(currentSegment)+(l/2)-1)>L
                    % segmento muito a direita
                    x_seg{currentSegment,currentChannel} = ...
                        x(finalCenterLocs(currentSegment)-l/2:L, ...
                        currentChannel);
                else
                    x_seg{currentSegment,currentChannel} = ...
                        x(finalCenterLocs(currentSegment)-l/2: ...
                        finalCenterLocs(currentSegment)+(l/2)-1, ...
                        currentChannel);
                end
            end
            case 1 % se l for impar
                if(finalCenterLocs(currentSegment) - (l-1)/2)<1
                    % segmento muito a esquerda
                    x_seg{currentSegment,currentChannel} = ...
                        x(1:finalCenterLocs(currentSegment) + (l-1)/2, ...
                        currentChannel);
                else if(finalCenterLocs(currentSegment) + (l-1)/2)>L
                    % segmento muito a direita

```

```
x_seg{currentSegment,currentChannel} = ...
x(finalCenterLocs(currentSegment) - (l-1)/2:L, ...
currentChannel);
else
x_seg{currentSegment,currentChannel} = ...
x(finalCenterLocs(currentSegment) - (l-1)/2: ...
finalCenterLocs(currentSegment) + (l-1)/2, ...
currentChannel);
end
end
end
end
end
```

# APÊNDICE C – Função em MATLAB para MTD3

```

function [x_seg, finalCenterLocs] = seg_mtd3(x, l_min, l_max, step, W, B, C)
% MTD3 - metodo com janela deslizante para deteccao de BEP e EEP de segmentos
% utilizando variacao total

%
% Argumentos: (para mais detalhes, refira a descricao do MTD3)
%   x - matriz cujas colunas sao canais do sinal a ser segmentado
%   l_min - compimento minimo para segmentos
%   l_max - compimento maximo para segmentos
%   step - numero de amostras a incrementar posicao de janela
%   W - comprimento da janela deslizante utilizada pelo metodo
%   B - valor limite para variacao total que determina um BEP
%   C - valor limite para variacao total que determina um EEP
%
% Retorno:
%   x_seg - cell array com os canais segmentados
%   finalCenterLocs - posicoes centrais dos segmentos

%% Preprocessamento
[L, numberOfChannels] = size(x); % comprimento do sinal e numero de canais
x_ret = abs(x); % retificacao
x_norm = zeros(L, numberOfChannels); % normalizacao
for currentChannel = 1:numberOfChannels
    x_norm(:,currentChannel) = ...
        x_ret(:,currentChannel)./max(x_ret(:,currentChannel));
end

%% Metodo

totalVariation = zeros(L-W,numberOfChannels);
BEPsLocsFlags = false(W,numberOfChannels);
EEPsLocsFlags = false(W,numberOfChannels);
BEPsLocsCell = cell(1,numberOfChannels);
EEPsLocsCell = cell(1,numberOfChannels);
searchBEP = true(numberOfChannels,1);
lastBEPloc = zeros(numberOfChannels,1);
for w0 = 1:step:L-W % janela deslizante para calculo de variacao total
    for currentChannel = 1:numberOfChannels
        totalVariation(w0, currentChannel) =...
            sum(diff(x_norm(w0:w0+W, currentChannel)));
    end
    if totalVariation(w0, currentChannel) >= B
        BEPsLocsFlags(w0) = true;
        BEPsLocsCell{w0} = currentChannel;
    end
    if totalVariation(w0, currentChannel) <= C
        EEPsLocsFlags(w0) = true;
        EEPsLocsCell{w0} = currentChannel;
    end
end
finalCenterLocs = BEPsLocsCell;

```

```

% Identificacao de BEPs e EEPs
switch searchBEP(currentChannel)
    case true % deteccao de BEPs
        if totalVariation(w0, currentChannel) > B
            BEPsLocsFlags(w0, currentChannel) = true;
            lastBEPloc(currentChannel) = w0;
            searchBEP(currentChannel) = false;
        end
    case false % deteccao de EEPs
        if (w0+W-lastBEPloc(currentChannel)) > l_max
            % segmento excederia comprimento maximo
            BEPsLocsFlags(lastBEPloc(currentChannel),currentChannel)=false;
            searchBEP(currentChannel) = true;
        else if (totalVariation(w0, currentChannel) < C) && ...
            (w0+W-lastBEPloc(currentChannel) > l_min)
            EEPsLocsFlags(w0+W,currentChannel) = true;
            searchBEP(currentChannel) = true;
        end
    end
end
for currentChannel = 1:numberOfChannels
    BEPsLocsCell{1,currentChannel} = find(BEPsLocsFlags(:,currentChannel));
    EEPsLocsCell{1,currentChannel} = find(EEPsLocsFlags(:,currentChannel));
end

%% Clustering

BEPsLocsArray = sort(cell2mat(BEPsLocsCell'));
EEPsLocsArray = sort(cell2mat(EEPsLocsCell'));
[~, labscoreBEPs] = dbscan(BEPsLocsArray,2000,3);
[~, labscoreEEPs] = dbscan(EEPsLocsArray,2000,3);
numberOfBEPs = max(labscoreBEPs);
numberOfEEPs = max(labscoreEEPs);
% medias internas aos clusters
meanBEPs = zeros(numberOfBEPs,1);
for currentCluster = 1:numberOfBEPs
    meanBEPs(currentCluster) = ...
        round(mean(BEPsLocsArray(labscoreBEPs == currentCluster)));
end
meanEEPs = zeros(numberOfEEPs,1);
for currentCluster = 1:numberOfEEPs
    meanEEPs(currentCluster) = ...
        round(mean(EEPsLocsArray(labscoreEEPs == currentCluster)));
end

```

```
%% Pareamento final de BEPs e EEPs
% (devem ocorrer alternadamente e atender requisitos de l_min e l_max)

allLocs = sortrows([meanBEPs,true(length(meanBEPs),1);...
    meanEEPs,false(length(meanEEPs),1)]);
locsDelta = diff(allLocs,1);
numberOfDeltas = length(locsDelta(:,1));
finalBEPsFlags = false(numberOfDeltas+1,1);
finalEEPsFlags = false(numberOfDeltas+1,1);
for currentDelta = 1:numberOfDeltas
    if (locsDelta(currentDelta,1) > l_min) && ...
        (locsDelta(currentDelta,1) < l_max) && ...
        locsDelta(currentDelta,2) == -1; % par BEP-EEP valido
        finalBEPsFlags(currentDelta) = true;
        finalEEPsFlags(currentDelta+1) = true;
    end
end
finalBEPs = allLocs(finalBEPsFlags,1);
finalEEPs = allLocs(finalEEPsFlags, 1);

%% Segmentacao

numberOfSegments = length(finalBEPs);
finalCenterLocs = zeros(numberOfSegments,1);
x_seg = cell(numberOfSegments,numberOfChannels);
for currentChannel = 1:numberOfChannels
    for currentSegment = 1:numberOfSegments
        x_seg{currentSegment,currentChannel} = ...
            x(finalBEPs(currentSegment):finalEEPs(currentSegment),currentChannel);
        finalCenterLocs(currentSegment) = ...
            round(mean([finalBEPs(currentSegment),finalEEPs(currentSegment)]));
    end
end
end
```

# APÊNDICE D – Função em MATLAB para MTD4

```

function [x_seg, finalCenterLocs] = seg_mtd4(x, l_min, l_max, step, W, T)
% MTD4 - metodo com janela deslizante para deteccao de BEP e EEP de segmentos
% utilizando threshold
%
% Argumentos: (para mais detalhes, refira a descricao do MTD3)
%   x - matriz cujas colunas sao canais do sinal a ser segmentado
%   l_min - compimento minimo para segmentos
%   l_max - compimento maximo para segmentos
%   step - numero de amostras a incrementar posicao de janela
%   W - comprimento da janela deslizante utilizada pelo metodo
%   T - valor de threshold para janela que determina BEPs e EEPs
%
% Retorno:
%   x_seg - cell array com os canais segmentados
%   finalCenterLocs - posicoes centrais dos segmentos

%% Preprocessamento
[L, numberOfChannels] = size(x); % comprimento do sinal e numero de canais
x_ret = abs(x); % retificacao
x_norm = zeros(L, numberOfChannels); % normalizacao
for currentChannel = 1:numberOfChannels
    x_norm(:,currentChannel) = ...
        x_ret(:,currentChannel)./max(x_ret(:,currentChannel));
end

%% Metodo

BEPsLocsFlags = false(W,numberOfChannels);
EEPsLocsFlags = false(W,numberOfChannels);
BEPsLocsCell = cell(1,numberOfChannels);
EEPsLocsCell = cell(1,numberOfChannels);
searchBEP = true(numberOfChannels,1);
lastBEPloc = zeros(numberOfChannels,1);
for w0 = 1:step:L-W % janela deslizante para calculo de variacao total
    for currentChannel = 1:numberOfChannels
        % Identificacao de BEPs e EEPs
        switch searchBEP(currentChannel)
            case true % deteccao de BEPs
                if max(x_norm(w0:w0+W, currentChannel)) > T

```

```

        BEPsLocsFlags(w0,currentChannel) = true;
        lastBEPloc(currentChannel) = w0;
        searchBEP(currentChannel) = false;
    end
case false % deteccao de EEPs
    if (w0+W-lastBEPloc(currentChannel)) > l_max
        % segmento excederia comprimento maximo
        BEPsLocsFlags(lastBEPloc(currentChannel),currentChannel)=false;
        searchBEP(currentChannel) = true;
    else if (max(x_norm(w0:w0+W, currentChannel)) < T) && ...
        (w0+W-lastBEPloc(currentChannel) > l_min)
        EEPsLocsFlags(w0+W,currentChannel) = true;
        searchBEP(currentChannel) = true;
    end
end
end
for currentChannel = 1:numberOfChannels
    BEPsLocsCell{1,currentChannel} = find(BEPsLocsFlags(:,currentChannel));
    EEPsLocsCell{1,currentChannel} = find(EEPssLocsFlags(:,currentChannel));
end

%% Clustering

BEPsLocsArray = sort(cell2mat(BEPsLocsCell'));
EEPssLocsArray = sort(cell2mat(EEPssLocsCell'));
[~, labscoreBEPs] = dbscan(BEPsLocsArray,2000,3);
[~, labscoreEEPs] = dbscan(EEPssLocsArray,2000,3);
numberOfBEPs = max(labscoreBEPs);
numberOfEEPs = max(labscoreEEPs);
% medias internas aos clusters
meanBEPs = zeros(numberOfBEPs,1);
for currentCluster = 1:numberOfBEPs
    meanBEPs(currentCluster) = ...
        round(mean(BEPsLocsArray(labscoreBEPs == currentCluster)));
end
meanEEPs = zeros(numberOfEEPs,1);
for currentCluster = 1:numberOfEEPs
    meanEEPs(currentCluster) = ...
        round(mean(EEPssLocsArray(labscoreEEPs == currentCluster)));
end

%% Pareamento final de BEPs e EEPs
% (devem ocorrer alternadamente e atender requisitos de l_min e l_max)

allLocs = sortrows([meanBEPs,true(length(meanBEPs),1);...

```

```
meanEEPs, false(length(meanEEPs),1));
locsDelta = diff(allLocs,1);
numberOfDeltas = length(locsDelta(:,1));
finalBEPsFlags = false(numberOfDeltas+1,1);
finalEEPsFlags = false(numberOfDeltas+1,1);
for currentDelta = 1:numberOfDeltas
    if (locsDelta(currentDelta,1) > l_min) && ...
        (locsDelta(currentDelta,1) < l_max) && ...
        locsDelta(currentDelta,2) == -1; % par BEP-EEP valido
        finalBEPsFlags(currentDelta) = true;
        finalEEPsFlags(currentDelta+1) = true;
    end
end
finalBEPs = allLocs(finalBEPsFlags,1);
finalEEPs = allLocs(finalEEPsFlags, 1);

%% Segmentacao

numberOfSegments = length(finalBEPs);
finalCenterLocs = zeros(numberOfSegments,1);
x_seg = cell(numberOfSegments,numberOfChannels);
for currentChannel = 1:numberOfChannels
    for currentSegment = 1:numberOfSegments
        x_seg{currentSegment,currentChannel} = ...
            x(finalBEPs(currentSegment):finalEEPs(currentSegment),currentChannel);
        finalCenterLocs(currentSegment) = ...
            round(mean([finalBEPs(currentSegment),finalEEPs(currentSegment)]));
    end
end
end
```

# APÊNDICE E – Código em MATLAB para segmentação de sinais, treino de RNA e resultados de classificação

```
% Implementacao dos metodos de segmentacao com variacao de parametros
% e resultados para calssificacao utilizando RNA
close all
clear

%% Selecao da base de dados (manter uma opcao comentada)
% Base de dados Ninapro
% database = 'ninapro';
% path = 'database/ninapro2/';
% subjectList = ls([path 'S*_E1*']);
% numberOfSubjects = 40;
% r_target = 5.6e-5;

% Base de dados IEE
database = 'iee';
path = 'database/IEE/';
subjectList = ls([path '*.mat']);
numberOfSubjects = 10;
r_target = 6.1e-5;

%% Selecao do metodo a ser utilizado e combinacoes a serem testadas
methodToTest = 1;
switch methodToTest
    case 1
        l = 10e3;
        q = [0.75 0.8 0.85 0.9 0.95];
        T_lim = [0.05 0.1 0.15 0.2 0.25];
        combinations = combvec(q, T_lim)';
        combArg1 = combinations(:,1);
        combArg2 = combinations(:,2);
    case 2
        l = 10e3;
        A = [60 80 100];
        B = [2 5 8];
        C = [2 5 8];
        combinations = combvec(A, B, C)';
    end
end
```

```

        combArg1 = combinations(:,1);
        combArg2 = combinations(:,2);
        combArg3 = combinations(:,3);

    case 3
        l_min = 7.5e3;
        l_max = 12.5e3;
        step = 100;
        W = 5e3;
        B = 0.05:0.05:0.25;
        C = -0.05:-0.05:-0.25;
        combinations = combvec(B, C)';
        combArg1 = combinations(:,1);
        combArg2 = combinations(:,2);

    case 4
        l_min = 7.5e3;
        l_max = 12.5e3;
        step = 100;
        W = 5e3;

        combinations = (0.01:0.01:0.25)';

end

%% Implementacao
numberOfClasses = 17;
numberOfCombinations = 25;
numberOfChannels = 12;
classificationOutput = cell(numberOfCombinations, numberOfSubjects);
targetsOutput = cell(numberOfCombinations, numberOfSubjects);
predictorsOutput = cell(numberOfCombinations, numberOfSubjects);

parfor_progress(numberOfSubjects);
for currentSubject = 1:numberOfSubjects
    parfor_progress; % exibe progresso do parfor
    S = load ([path subjectList(currentSubject,:)]);
    L = length(S.emg);
    x = S.emg;
    stimulus = S.stimulus;

    parfor currentCombination = 1:numberOfCombinations
        % metodo de segmentacao
        switch methodToTest
            case 1
                [x_seg, centerLocs] = ...
                    seg_mtd1(x, l, combArg1(currentCombination), r_target, ...
                    combArg2(currentCombination));
            case 2
                [x_seg, centerLocs] = ...

```

```

        seg_mtd2(x, l, combArg1(currentCombination), ...
        combArg2(currentCombination), combArg3(currentCombination));
case 3
    [x_seg, centerLocs] = ...
        seg_mtd3(x, l_min, l_max, step, W, ...
        combArg1(currentCombination), combArg2(currentCombination));
case 4
    [x_seg, centerLocs] = ...
        seg_mtd4(x, l_min, l_max, step, W, T(currentCombination));
end
% identificacao do movimento correspondente a cada segmento
if strcmp(database, 'ninapro')
    targetClasses = identifyClasses(centerLocs, database, stimulus);
else
    targetClasses = identifyClasses(centerLocs, database, L);
end
targetsOutput{currentCombination, currentSubject} = targetClasses;

% Divisao de grupos para treinamento
numberOfSegments = length(centerLocs);
numberOfTrainPerClass = sum(targetClasses,1);
numberOfTrainPerClass(numberOfTrainPerClass == 2) = ...
    numberOfTrainPerClass(numberOfTrainPerClass == 2) - 1;
numberOfTrainPerClass(numberOfTrainPerClass > 2) = ...
    numberOfTrainPerClass(numberOfTrainPerClass > 2) - 2;

trainIndFlags = false(numberOfSegments,1);
valIndFlags = false(numberOfSegments,1);
testIndFlags = false(numberOfSegments,1);
for currentClass = 1:numberOfClasses
    counter = 0;
    for currentSegment = 1:numberOfSegments
        if targetClasses(currentSegment,currentClass)
            if counter < numberOfTrainPerClass(currentClass)
                trainIndFlags(currentSegment) = true;
            else if counter == numberOfTrainPerClass(currentClass)
                valIndFlags(currentSegment) = true;
            else if counter > numberOfTrainPerClass(currentClass)
                testIndFlags(currentSegment) = true;
            end
        end
        counter = counter + 1;
    end
end
trainInd = find(trainIndFlags);

```

```
valInd = find(valIndFlags);
testInd = find(testIndFlags);

% Preditores da RNA
predictors = zeros(numberOfSegments, 3*numberOfChannels);
for currentSegment = 1:numberOfSegments
    for currentChannel = 1:numberOfChannels
        % RMS
        predictors(currentSegment, currentChannel + 0*numberOfChannels) = ...
            rms(x_seg{currentSegment}, currentChannel));
        % Variância
        predictors(currentSegment, currentChannel + 1*numberOfChannels) = ...
            var(x_seg{currentSegment}, currentChannel));
        % Frequência mediana
        predictors(currentSegment, currentChannel + 2*numberOfChannels) = ...
            medfreq(x_seg{currentSegment}, currentChannel));
    end
end
predictorsOutput{currentCombination, currentSubject} = predictors;

% Treinamento de rede neural
net = patternnet(40, 'trainscg');
net.divideFcn = 'divideind';
net.divideParam.trainInd = trainInd;
net.divideParam.valInd = valInd;
net.divideParam.testInd = testInd;
trainedNet = train(net, predictors', targetClasses');
classificationOutput{currentCombination, currentSubject} = ...
    trainedNet(predictors'); % resultados de classificação
end
S = []; % libera espaço da memória
end
save(['./out/workspace/MTD' num2str(methodToTest) '_' database '.mat'])
```

# APÊNDICE F – Código utilizado para determinação de $r_{target}$ do MTD1

```
% Determinacao de r_target para MTD1

%% Base de dados Ninapro
ninaproList = ls('database/ninapro2/S*_E1*');
numberOfSubjects = length(ninaproList);
numberOfSegments = 102; % 17 movimentos * 6 repeticoes
r_target = zeros(numberOfSubjects, 1);
for currentSubject = 1:numberOfSubjects
    fprintf('currentSubject = %i\n', currentSubject)
    % Carrega o voluntario atual
    load ([ 'database/ninapro2/' ninaproList(currentSubject, :) ])
    % r_target para este voluntario
    r_target(currentSubject) = numberOfSegments/length(emg);
end
min_r_target(1) = min(r_target);

%% Base de dados IEE
ieeList = ls('database/IEE/*.mat');
numberOfSubjects = length(ieeList(:,1));
numberOfSegments = 102; % 17 movimentos * 6 repeticoes
r_target = zeros(numberOfSubjects, 1);
for currentSubject = 1:numberOfSubjects
    fprintf('currentSubject = %i\n', currentSubject)
    % Carrega o voluntario atual
    load ([ 'database/IEE/' ieeList(currentSubject, :) ])
    % r_target para este voluntario
    r_target(currentSubject) = numberOfSegments/length(emg);
end
min_r_target(2) = min(r_target);

% Resultados:
%   min_r_target(1) = 5.6241e-5 [NinaPro]
%   min_r_target(2) = 6.1874e-5 [IEE]
```

# APÊNDICE G – Código utilizado para escolha de valores de $A$ do MTD2

```
% Determinacao de A para MTD2

%% Base de dados Ninapro
ninaproList = ls('database/ninapro2/S*_E1*');
numberOfSubjects = length(ninaproList);
numberOfChannels = 12; % 17 movimentos * 6 repeticoes
A_value = zeros(numberOfSubjects, numberOfChannels);
for currentSubject = 1:numberOfSubjects
    fprintf('currentSubject = %i\n', currentSubject)
    load ([ 'database/ninapro2/' ninaproList(currentSubject,:) ])
    [L, ~] = size(emg); % comprimento do sinal e numero de canais
    x_ret = abs(emg); % retificacao
    x_norm = zeros(L, numberOfChannels); % normalizacao
    for currentChannel = 1:numberOfChannels
        x_norm(:,currentChannel) = ...
            x_ret(:,currentChannel)./max(x_ret(:,currentChannel));
    end
    % Razao entre valor maximo e medio para canais deste voluntario
    A_value(currentSubject,:) = 1./mean(x_norm);
end
% mediana de razoes para todos os voluntarios e canais
med_Avalue(1) = median(median(A_value));

%% Base de dados IEE
ieeList = ls('database/IEE/*.mat');
numberOfSubjects = length(ieeList(:,1));
numberOfChannels = 12; % 17 movimentos * 6 repeticoes
A_value = zeros(numberOfSubjects, numberOfChannels);
for currentSubject = 1:numberOfSubjects
    fprintf('currentSubject = %i\n', currentSubject)
    load ([ 'database/ninapro2/' ninaproList(currentSubject,:) ])
    [L, ~] = size(emg); % comprimento do sinal e numero de canais
    x_ret = abs(emg); % retificacao
    x_norm = zeros(L, numberOfChannels); % normalizacao
    for currentChannel = 1:numberOfChannels
        x_norm(:,currentChannel) = ...
            x_ret(:,currentChannel)./max(x_ret(:,currentChannel));
    end
    % Razao entre valor maximo e medio para canais deste voluntario
```

```
A_value(currentSubject,:)=1./mean(x_norm);  
end  
% mediana de razoes para todos os voluntarios e canais  
med_Avalue(2)=median(median(A_value));  
  
% Resultados:  
% med_Avalue(1)=73.8127 [NinaPro]  
% med_Avalue(2)=83.3154 [IEE]
```

# APÊNDICE H – Função em MATLAB para obtenção de resposta esperada de treino de RNA

```

function targetdataArray = identifyClasses(centerLocs, database, varargin)

if strcmp(database, 'ninapro')
    %% Remodela o vetor de estimulo, removendo trechos de 0
    stimulus = varargin{1};
    classLabel = stimulus;
    currentClass = 1;
    for index = 1:length(stimulus)
        if classLabel(index) ~= 0
            currentClass = classLabel(index);
        else
            classLabel(index) = currentClass;
        end
    end
end
if strcmp(database, 'iee')
    %% Divide temporalmente o sinal em 17 partes
    L = varargin{1};
    classLabel = zeros(L,1);
    for currentClass = 1:L
        for index = 1:L
            if index <= currentClass*L/17 && index > (currentClass-1)*L/17
                classLabel(index) = currentClass;
            end
        end
    end
end
%% Gera a matriz de identificacao das classes
numberOfSegments = length(centerLocs);
numberOfClasses = 17;
targetdataArray = false(numberOfSegments, numberOfClasses);
for currentSegment = 1:numberOfSegments
    targetdataArray(currentSegment, ...
        classLabel(centerLocs(currentSegment))) = true;
end
end

```

## Anexos

# ANEXO A – Função em MATLAB para agrupamento com DBSCAN (TRAN; DRAB; DASZYKOWSKI, 2013)

```

function [labs labscore] = dbscan(a,Eps,MinPts)
% DBSCAN clustering
% [labs labscore] = dbscan(A,Eps,MinPts)
%
% DBSCAN clustering of data matrix in A. labels is a vector with
% cluster labels for each vector.
%
% In case of publication of any application of this method,
% please, cite the original work:
% Thanh N. Tran*, Klaudia Drab, Michal Daszykowski, "Revised DBSCAN algorithm
% to cluster data with dense adjacent clusters", Chemometrics and Intelligent
% Laboratory Systems, 120:92–96.
% DOI: 10.1016/j.chemolab.2012.11.006

UNCLASSIFIED = 0;
BORDER = -2;
% Square Eps in order not to square all distances of points
%eps = eps^2;

m = size(a,1);
labs = zeros(m,1);
ClusterId = 1;
for i=1:m
    if labs(i) == UNCLASSIFIED

        % Expand cluster ClusterId
        % Get a set of points of distance < eps
        [ExpandClusterReturn labs]= ...
            ExpandCluster(a,labs,i,ClusterId,Eps, MinPts);
        if ExpandClusterReturn
            ClusterId = ClusterId +1;
        end
    end
end

```

```
% Step 3:  
labscore = labs; core_index = find(labscore > 0);  
border_points = find(labs==BORDER);  
% For xborder in border_list but has no ClusterId  
for i=1:length(border_points)  
    % xborder in border_list  
    currentB = border_points(i);  
    d=distance(+a(currentB,:),+a(core_index,:),1);  
    % the closest core-points  
    [tmp nearest_core]=min(d);  
    nearest_core_index=core_index(nearest_core);  
    %Assign xborder to ClusterId of the closest core-points  
    labs(currentB)=labs(nearest_core_index);  
end  
end  
  
function [ExpandClusterReturn labs]= ...  
    ExpandCluster(a,labs,i,ClusterId, Eps, MinPts)  
UNCLASSIFIED = 0;  
NOISE = -1;  
BORDER = -2;  
% calculate distances  
d=distance(+a(i,:),+a(:,:,1),1);  
  
% seeds = Retrieve_Neighbors(xi, Eps)  
seeds = find(d < Eps);  
  
% If |seeds| < MinPts  
if size(seeds,2) < MinPts,  
    labs(i) = NOISE; % Assign xi as noise  
    ExpandClusterReturn = 0; % Return without expansion success  
else  
    % STEP 1: xi is identified as a starting core-point for ClusterId  
    %labs(i) = ClusterId; % Thanh changed in May 2012  
    % Exclude xi from the Seeds  
    %seeds = setdiff(seeds, i); % Thanh changed in May 2012  
    % STEP 2: Identify chains  
    % For all xj in seeds  
    while ~isempty(seeds) % Not an empty seeds  
        % current point is the first point in seeds  
  
        currentP = seeds(1);  
        d=distance(+a(currentP,:),+a(:,:,1),1);  
        % NEps(xj)= Retrieve_Neighbors(xj,Eps)  
        result = find(d <= Eps);  
        %If |NEps(xj)| >= MinPts // xj is a core point  
        if length(result) >= MinPts
```

```
% Assign xj to ClusterId
labs(currentP) = ClusterId;
% Add all UNCLASSIFIED in NEps(xj) to seeds
result_unclassified = result(find(labs(result)==UNCLASSIFIED));
result_noise = result(find(labs(result)==NOISE));
% Temporary complete the intermediate chain ...
% the chain can be extended in the comming steps.
% The border points have not yet been assigned to any cluster

% first assign to a border-point...
% later will be reassigned to e.g. core-point
labs([result_unclassified result_noise]) = BORDER;
seeds = union(seeds,result_unclassified);

end
% Exclude the current point in seeds and go back to the loop
seeds = seeds(2:size(seeds,2));
end % end while
% Return with expansion success
ExpandClusterReturn = 1;      % return true
end
end
```